

# A QUANTUM BEHAVED PARTICLE SWARM APPROACH TO MULTI-OBJECTIVE OPTIMIZATION

by

HEYAM AL-BAITY

A thesis submitted to  
The University of Birmingham  
for the degree of  
DOCTOR OF PHILOSOPHY

School of Computer Science  
College of Engineering and Physical Sciences  
The University of Birmingham  
30. april 2015

UNIVERSITY OF  
BIRMINGHAM

**University of Birmingham Research Archive**

**e-theses repository**

This unpublished thesis/dissertation is copyright of the author and/or third parties. The intellectual property rights of the author or third parties in respect of this work are as defined by The Copyright Designs and Patents Act 1988 or as modified by any successor legislation.

Any use made of information contained in this thesis/dissertation must be in accordance with that legislation and must be properly acknowledged. Further distribution or reproduction in any format is prohibited without the permission of the copyright holder.



---

## Abstract

---

Many real-world optimization problems have multiple objectives that have to be optimized simultaneously. In multi-objective optimization problems, the major challenge is to find the set of solutions that achieve the best compromise with regard to the whole set of objectives. Although a great deal of effort has been devoted to solving multi-objective optimization problems, the problem is still open and the related issues still attract significant research efforts. The possibility to get a set of Pareto optimal solutions in a single run is one of the attracting and motivating features of using population based algorithms to solve optimization problems with multiple objectives. Most of the proposed approaches make use of metaheuristics. Their basic idea is to introduce the Pareto dominance concept into nature inspired algorithms such as Genetic Algorithms (GAs) and Particle Swarm Optimization (PSO). Quantum-behaved Particle Swarm Optimization (QPSO) is a recently proposed population based metaheuristic that relies on quantum mechanics principles. Since its inception, much effort has been devoted to developing improved versions or new applications of QPSO designed for single objective optimization. However, many of its advantages are not yet available for multi-objective optimization. In this thesis, we develop a new framework for multi-objective problems using QPSO. The contribution of the work is threefold. First a hybrid leader selection method has been developed to compute the attractor of a given particle and applied in unconstrained optimization case. Its aim is to foster convergence of the obtained Pareto fronts while maintaining good diversity. Second, an archiving strategy has been proposed to control the growth of the archive size

in order to achieve a balance between the quality of solutions of an unbounded archive method and the cost effectiveness of a bounded archive method. Third, the developed framework has been further extended to handle constrained optimization problems. A comprehensive investigation of the developed framework has been carried out under different selection, archiving and constraint handling strategies. The developed framework is found to be a competitive technique to tackle this type of problems when compared against the state-of-the-art methods in multi-objective optimization.

---

## Acknowledgements

---

First and foremost, I owe my deepest gratitude to my supervisors Souham Meshoul and Ata Kaban.

I would like to thank and acknowledge the continued motivation, unlimited patience, and endless support Dr. Souham has given me during the period of my study. This thesis would not have been possible without her help and guidance.

My heartfelt appreciation and acknowledgment to Dr. Ata for her advice, time, and valuable and constructive feedback.

I would also like to acknowledge the support given by King Saud University through a scholarship to pursue my degree at the University of Birmingham, and not to forget Dr. Lilac Alsafadi, my co-internal supervisor.

I am forever thankful to my family for their constant support. Lastly, my special thanks to my husband Ahmad Al-Saif for his time, patience, and unconditional love and support.



---

# Indhold

---

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Context of the Work . . . . .	1
1.2	Motivation . . . . .	3
1.3	Research Questions . . . . .	6
1.4	Contributions . . . . .	7
1.5	Publications . . . . .	7
1.6	Thesis Outline . . . . .	8
<b>2</b>	<b>Background and Review of Related Work</b>	<b>11</b>
2.1	Basic Concepts of Multi-objective Optimization . . . . .	11
2.1.1	Optimization Problems . . . . .	11
2.1.2	Single Objective Optimization Problems . . . . .	12
2.1.3	Multi Objective Optimization Problems . . . . .	15
2.1.4	Computational Complexity Considerations . . . . .	22
2.1.5	Why Metaheuristics for Continuous MOPs . . . . .	24
2.2	Introduction to Swarm Based Metaheuristics . . . . .	25
2.2.1	Particle Swarm Optimization . . . . .	27
2.2.2	From PSO to QPSO . . . . .	32
2.3	Review of the Past Related Work . . . . .	36
2.3.1	Weighted Sum Aggregation Function Based Approaches . . . . .	37
2.3.2	Lexicographic Ordering Based Approaches . . . . .	37
2.3.3	Non-Pareto Vector Evaluated Approaches . . . . .	38
2.3.4	Pareto-based EAs Approaches . . . . .	40
2.4	Inadequacies of Previous Work . . . . .	56
2.5	Summary . . . . .	59
<b>3</b>	<b>A New Framework for MOP: Multi-Objective Quantum-behaved Particle Swarm Optimization (MOQPSO) for Unconstrained Problems</b>	<b>61</b>
3.1	Main Features of the Proposed MOQPSO . . . . .	62
3.2	Description of the Proposed MOQPSO . . . . .	64
3.2.1	Computing Local Attractor Points in MOP . . . . .	64
3.2.2	Calculating the Mean Best Position . . . . .	67
3.2.3	Setting of the Contraction Expansion Parameter $\beta$ . . . . .	67



3.2.4	Outline of the General Framework of MOQPSO for Unconstrained Problems . . . . .	67
3.3	The Proposed Leader Selection Strategy . . . . .	70
3.4	Archiving Mechanism . . . . .	73
3.5	Time Complexity Analysis of MOQPSO for Unconstrained Problems . . . . .	75
3.6	Evaluation of the Proposed MOQPSO Algorithm . . . . .	77
3.6.1	Test Functions . . . . .	78
3.6.2	Performance Measures . . . . .	80
3.6.3	Parameter Settings . . . . .	82
3.6.4	Experimental Results and Discussions . . . . .	84
3.7	Summary . . . . .	88
<b>4</b>	<b>Constraint Handling in MOQPSO</b>	<b>89</b>
4.1	Constrained Multi-objective Optimization Problems (CMOPs) . . . . .	89
4.2	The Proposed MOQPSO for CMOPs . . . . .	91
4.2.1	CMOQPSO with the First Constraint Handling Strategy . . . . .	91
4.2.2	CMOQPSO with the Second Constraint Handling Strategy . . . . .	92
4.3	Time Complexity Analysis of CMOQPSO for Constrained Problems . . . . .	98
4.4	Experiments . . . . .	99
4.4.1	Experimental Set up . . . . .	102
4.4.2	Performance Measures . . . . .	103
4.4.3	Experimental Results and Discussions . . . . .	103
4.4.4	Study of the Effect of the Probabilistic Selection Rules of <i>gbest</i> and <i>sbest</i> positions . . . . .	108
4.5	Summary . . . . .	117
<b>5</b>	<b>An Extensive Empirical Comparison of Different Selection Strategies for Constrained and Unconstrained Problems</b>	<b>119</b>
5.1	Experiments . . . . .	120
5.1.1	<b>Experiment1:</b> Impact of Leader Selection Strategies . . . . .	120
5.1.2	<b>Experiment2:</b> MOQPSO vs MOPSO . . . . .	141
5.2	Summary . . . . .	145
<b>6</b>	<b>An Extensive Empirical Study of the Impact of Different Archiving Strategies for Constrained and Unconstrained MOPs</b>	<b>147</b>
6.1	Handling Archive Size . . . . .	148
6.1.1	The Proposed Redundancy Removal Archiving Strategy . . . . .	149
6.2	Time Complexity Analysis of the Archiving Methods . . . . .	151

6.3	<b>Experiment:</b> Impact of archiving methods . . . . .	153
6.4	Summary . . . . .	178
<b>7</b>	<b>MOQPSO-Clust: An Application of MOQPSO to Clustering</b>	<b>181</b>
7.1	Overview on Data Clustering . . . . .	182
7.2	Why Multi-objective Clustering . . . . .	184
7.3	Clustering as a MOP . . . . .	184
7.3.1	Related Work . . . . .	185
7.4	The Proposed MOQPSO for Clustering (MOQPSO-Clust) . . . . .	186
7.4.1	Problem formulation . . . . .	186
7.4.2	Cluster Encoding . . . . .	187
7.4.3	The objective Functions . . . . .	189
7.4.4	Outline of MOQPSO for Clustering (MOQPSO-Clust) . . . . .	190
7.5	Time Complexity Analysis of MOQPSO-Clust for Data Clustering . . . . .	192
7.6	Experiments . . . . .	194
7.6.1	Experimental Set Up . . . . .	196
7.6.2	Experimental Results and Discussions . . . . .	197
7.7	Summary . . . . .	204
<b>8</b>	<b>Conclusions and Future Work</b>	<b>205</b>
8.1	Summary of the Work . . . . .	205
8.2	Future Work . . . . .	211
	<b>Bibliography</b>	<b>215</b>



---

## Figurer

---

2.1	Types of optimization problems . . . . .	13
2.2	Simple example of a MOP inspired by [31] . . . . .	17
2.3	Example of a Pareto front and dominated solutions in the objective space for a minimization problem. Taken from [31]. . . . .	19
2.4	Difference between SOP and MOP inspired by [31]. . . . .	20
2.5	Mapping between decision space and objective space taken from [31]. . . .	21
2.6	Taxonomy of multi-objective evolutionary algorithms . . . . .	36
3.1	The key components in MOQPSO design . . . . .	63
3.2	Selecting global leader from GBA for k=4: filled circles are GBA members, empty circle represents particle P . . . . .	71
3.3	The true Pareto optimal and obtained fronts of the used test functions . .	87
4.1	The true Pareto optimal and the obtained fronts of SRN and MOBES test functions using the first strategy (left column) and the second strategy (right column). . . . .	105
4.2	The true Pareto optimal and the obtained fronts of KITA, CONSTR, and TNK test functions using the first strategy (left column) and the second strategy (right column). . . . .	106
5.1	Graphs of multicomparison tests of selection methods based on Friedman statistics on convergence values for unconstrained functions. Overlapping intervals indicate no significant difference . . . . .	129
5.2	Graphs of multicomparison tests of selection methods based on Friedman statistics on diversity values for unconstrained functions. Overlapping intervals indicate no significant difference . . . . .	130
5.3	Box plots of CPU time for unconstrained functions with four selection methods based on 1. Proposed hybrid method, 2. Crowding distance, 3. Sigma method, 4. Random selection. . . . .	131
5.4	Graphs of multicomparison tests of selection methods based on Friedman statistics on convergence values for the first constraint strategy. Overlapping intervals indicate no significant difference. . . . .	138

5.5	Graphs of multicomparison tests of selection methods based on Friedman statistics on convergence values for the second constraint strategy. Overlapping intervals indicate no significant difference. . . . .	139
5.6	Box plots of CPU time on constrained functions with four selection methods based on 1.Hybrid method, 2.Crowding distance, 3.Sigma method, 4.Random selection. First constraint strategy (left column) and second constraint strategy (right column). . . . .	140
6.1	Box plots of CPU time for unconstrained functions with five archiving methods based on 1. Unbounded archive, 2. Clustering, 3. Crowding, 4. Maximin, and 5. Redundancy removal. . . . .	159
6.2	Graphs of multicomparison tests of archiving methods based on Friedman statistics on convergence values for unconstrained functions. Overlapping intervals indicate no significant difference . . . . .	161
6.3	Graphs of multicomparison tests of archiving methods based on Friedman statistics on diversity values for unconstrained functions. Overlapping intervals indicate no significant difference . . . . .	162
6.4	Graphs of multicomparison tests of archiving methods based on Friedman statistics on convergence values for the first constraint strategy. Overlapping intervals indicate no significant difference. . . . .	164
6.5	Graphs of multicomparison tests of archiving methods based on Friedman statistics on diversity values for the first constraint strategy. Overlapping intervals indicate no significant difference. . . . .	167
6.6	Box plots of CPU time for constrained functions with first constraint strategy on five archiving methods based on 1. Unbounded, 2. Clustering, 3. Crowding, 4. Maximin, and 5. Redundancy removal. . . . .	168
6.7	Graphs of multicomparison tests of archiving methods based on Friedman statistics on convergence values for the second constraint strategy. Overlapping intervals indicate no significant difference. . . . .	170
6.8	Graphs of multicomparison tests of archiving methods based on Friedman statistics on diversity values for the second constraint strategy. Overlapping intervals indicate no significant difference. . . . .	172
6.9	Box plots of CPU time for constrained functions with second constraint strategy on five archiving methods based on 1. Unbounded, 2. Clustering, 3. Crowding, 4. Maximin, and 5. Redundancy removal. . . . .	173
6.10	Archive size growth of the five archiving methods for unconstrained functions.	175
6.11	Archive size growth of the five archiving methods for constrained functions with the first constraint handling strategy. . . . .	176
6.12	Archive size growth of five archiving methods for constrained functions with the second constraint handling strategy. . . . .	177
7.1	dataset of 5 points with 3 attributes. . . . .	188
7.2	Encoding of particle's position when the number of clusters is $k=2$ . . . . .	188

7.3 Pareto solutions representing the trade-off partitions for 2d4c dataset (the first three solutions). . . . . 201

7.4 Pareto solutions representing the trade-off partitions for 2d4c dataset (the remaining six solutions). . . . . 202



---

## Tabeller

---

2.1	SOP vs. MOP . . . . .	20
3.1	Comparison between a single-objective and multi-objective QPSO optimizations . . . . .	65
3.2	Convergence and diversity values of the used test functions with different ranges of parameter $\beta$ . . . . .	83
3.3	Parameter settings of MOQPSO for unconstrained test problems. . . . .	83
3.4	Comparison with other MOEA: Convergence results. Mean (first row) and variance (second row) . . . . .	86
3.5	Comparison with other MOEA: Diversity results. Mean (first row) and variance (second row) . . . . .	86
4.1	Parameter settings of CMOQPSO with first and second constraint handling strategies for the constraint test functions. . . . .	103
4.2	Convergence and diversity values of the used test functions for the first and the second constraint handling strategies . . . . .	107
4.3	The different combinations of infeasible <i>sbest</i> and <i>gbest</i> used in this study	108
4.4	Statistical results of convergence metric with 30% favouring infeasible <i>sbest</i>	110
4.5	Statistical results of diversity metric with 30% favouring infeasible <i>sbest</i> . .	111
4.6	Statistical results of convergence metric with 50% favouring infeasible <i>sbest</i>	112
4.7	Statistical results of diversity metric with 50% favouring infeasible <i>sbest</i> . .	113
4.8	Statistical results of convergence metric with 70% favouring infeasible <i>sbest</i>	114
4.9	Statistical results of diversity metric with 70% favouring infeasible <i>sbest</i> . .	115
4.10	Statistical results of convergence metric with 0% favouring infeasible <i>sbest</i>	116
4.11	Statistical results of diversity metric with 0% favouring infeasible <i>sbest</i> . .	117
5.1	Parameter settings of MOQPSO for experiments to compare different selection methods on unconstrained test problems. . . . .	122
5.2	Parameter settings of CMOQPSO with the first and the second constraint handling strategies for experiments to compare different selection methods.	122



5.3	Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to convergence results on unconstrained functions (FON, SCH, ZDT1) with four selection methods based on 1.Hybrid method, 2.Crowding distance, 3.Sigma method, 4.Random selection. . . . .	125
5.4	Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to convergence results on unconstrained functions (ZDT2, ZDT3, ZDT6) with four selection methods based on 1.Hybrid method, 2.Crowding distance, 3.Sigma method, 4.Random selection. . . . .	126
5.5	Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to diversity results on unconstrained functions (FON, SCH, ZDT1) with four selection methods based on 1.Hybrid method, 2.Crowding distance, 3.Sigma method, 4.Random selection. . . . .	127
5.6	Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to diversity results on unconstrained functions (ZDT2, ZDT3, ZDT6) with four selection methods based on 1.Hybrid method, 2.Crowding distance, 3.Sigma method, 4.Random selection. . . . .	128
5.7	Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to convergence results for constrained functions with first constraint strategy on four selection methods based on 1.Hybrid method, 2.Crowding distance, 3.Sigma method, 4.Random selection. . . . .	134
5.8	Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to diversity results for constrained functions with first constraint strategy on four selection methods based on 1.Hybrid method, 2.Crowding distance, 3.Sigma method, 4.Random selection. . . . .	135
5.9	Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to convergence results for constrained functions with second constraint strategy on four selection methods based on 1.Hybrid method, 2.Crowding distance, 3.Sigma method, 4.Random selection. . . . .	136
5.10	Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to diversity results for constrained functions with second constraint strategy on four selection methods based on 1.Hybrid method, 2.Crowding distance, 3.Sigma method, 4.Random selection. . . . .	137
5.11	Parameter settings of MOPSO and MOQPSO for unconstrained test problems. . . . .	142
5.12	Parameter settings of MOPSO and MOQPSO with second constraint handling strategy for the constraint test functions. . . . .	142
5.13	Results of MOPSO and MOQPSO for constrained test functions in terms of statistics on convergence, diversity and the p-value with respect to ( $p < \alpha = 0.05$ ). . . . .	143
5.14	Results of MOPSO and MOQPSO for unconstrained test functions in terms of statistics on convergence, diversity and the p-value with respect to ( $p < \alpha = 0.05$ ). . . . .	144

6.1	Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to convergence results on unconstrained functions (FON, SCH, ZDT1) with five archiving methods numbered as 1. Unbounded, 2. Clustering, 3. Crowding 4. Maximin and 5. Redundancy removal . . . . .	154
6.2	Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to convergence results on unconstrained functions (ZDT2, ZDT3, ZDT6) with five archiving methods numbered as 1. Unbounded, 2. Clustering, 3. Crowding 4. Maximin and 5. Redundancy removal . . . . .	155
6.3	Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to diversity results on unconstrained functions (FON, SCH, ZDT1) with five archiving methods numbered as 1. Unbounded, 2. Clustering, 3. Crowding 4. Maximin and 5. Redundancy removal. . . . .	156
6.4	Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to diversity results on unconstrained functions (ZDT2, ZDT3, ZDT6) with five archiving methods numbered as 1. Unbounded, 2. Clustering, 3. Crowding 4. Maximin and 5. Redundancy removal. . . . .	157
6.5	Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to convergence results for constrained functions with first constraint strategy on five archiving methods based on 1. Unbounded, 2. Clustering, 3. Crowding, 4. Maximin, and 5. Redundancy removal. . . . .	163
6.6	Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to diversity results for constrained functions with first constraint strategy on five archiving methods based on 1. Unbounded, 2. Clustering, 3. Crowding, 4. Maximin, and 5. Redundancy removal. . . . .	166
6.7	Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to convergence results for constrained functions with second constraint strategy on five archiving methods based on 1. Unbounded, 2. Clustering, 3. Crowding, 4. Maximin, and 5. Redundancy removal. . . . .	169
6.8	Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to diversity results for constrained functions with second constraint strategy on five archiving methods based on 1. Unbounded, 2. Clustering, 3. Crowding, 4. Maximin, and 5. Redundancy removal. . . . .	171
7.1	Characteristics of the used synthetic datasets . . . . .	194
7.2	Characteristics of the used real datasets . . . . .	195
7.3	Obtained clustering results vs truth clustering results for real data sets . .	199
7.4	Obtained clustering results vs truth clustering results for synthetic data sets	200
7.5	F-measure results using MOQPSO-Clust and K-means on synthetic and real data sets. . . . .	203
7.6	Silhouette measure results using MOQPSO-Clust and K-means on synthetic and real data sets. . . . .	203
7.7	Comparison with other algorithms based on Median (Interquartile). The results in this table are quoted from [87]. . . . .	203



---

## Notations and Conventions

---

In the following list, we provide some definitions of the notations and acronyms used throughout the thesis:

- **MOP**: Multi-objective Optimization Problem.
- **SOP**: Single-objective Optimization Problem.
- **EA**: Evolutionary Algorithms.
- **MOEA**: Multi-Objective Evolutionary Algorithms.
- **NIC**: Nature Inspired Computing.
- **SI**: Swarm Intelligence.
- **PSO**: Particle Swarm Optimization.
- **QPSO**: Quantum-Behaved Particle Swarm Optimization.
- **MOPSO**: Multi-Objective Particle Swarm Optimization.
- **MOQPSO**: Multi-Objective Quantum-behaved Particle Swarm Optimization.
- **CMOP**: Constrained Multi-objective Optimization Problem.
- **CMOQPSO**: Constrained Multi-objective Quantum-behaved Particle Swarm Optimization.

- **MOQPSO-Clust**: Multi-Objective Quantum-behaved Particle Swarm Optimization for Clustering.
- **Swarm**: Population of candidate solutions in one generation of the evolutionary algorithm.
- **Particle**: One of the candidate solutions in the swarm.
- **selfbest**: Self best position of a given particle. it is the best position a given particle has reached so far.
- **gbest**: Global best position. It is the best position found so far by the whole swarm.
- **mbest**: Mean best position. It is the mean of self best positions of all particles.
- **GBA**: Global Best Archive. It is the archive of the non-dominated feasible solutions.
- **GBIA**: Global Best Infeasible Archive. It is the archive of the best infeasible solutions.
- **N**: Population size.
- **D**: Problem dimension.

## Introduction

---

### 1.1 Context of the Work

Optimization represents an important scientific field on its own. It is at the heart of many different disciplines ranging from science to industry and covering a number of areas like business, finance and economics to name just a few. In the computing field, optimization is concerned with the development of computational models, methods and tools that help making the best choice among a set of alternatives based on some criteria. An alternative represents a potential solution to the problem. The set of alternatives represents the search space and the criteria refer usually to objective functions. In its simplest form, an optimization problem has only one objective to be optimized. In this case, we deal with single objective optimization. For example, maximizing a profit function or minimizing a cost function requires finding the values of the problem's decision variables that give the best value of this function. A plethora of methods in the literature are devoted to solving single objective optimization problems. However, real-world problems are intrinsically multi-objective. Multi-objectivity adds a supplementary difficulty as the objectives are generally conflicting with each other, and should be optimized simultaneously. Such

problems are known as Multi-objective Optimization Problems (MOPs). These problems are challenged to determine balanced solutions that achieve the best compromise between objectives. Converting a MOP into a single objective optimization problem by aggregation of objectives was among the first attempts to solve MOPs. However, this is not always easy to do. Another alternative was to order objectives and accomplish a series of single objective optimizations. Generally, it is difficult, if not impossible, to establish such an ordering. The complexity of MOPs is such that efforts have been spent to design new methodologies based on nature inspired metaheuristics [31].

Nature Inspired Computing (NIC) is the field of developing new computing models that are inspired from natural systems. The reason is that natural systems exhibit important abilities such as learning, adaptation and optimization. Therefore, studying such systems in the quest to develop computational models is certainly a promising research area. NIC represents a paradigm shift in problem solving methodology. Nowadays, several nature inspired computational models exist such as evolutionary computation and swarm intelligence. Swarm intelligence deals with the collective behaviour within swarms such as colonies of ants and bees, flocks of birds and schools of fish. Within this context, several metaheuristics have been developed among which Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Artificial Bee Colony (ABC) and very recently Quantum Behaved PSO (QPSO) [121][21].

The subject of this thesis lies at the intersection of two fields: Optimization and Swarm Intelligence. In the former, the focus is on multi-objective optimization problems while in the latter we are interested in the QPSO model.

## 1.2 Motivation

When solving a multi-objective optimization problem, it is often not possible to improve one objective without deteriorating another conflicting one. Therefore, multi-objective optimization is challenged to determine the set of solutions that achieve the best compromise with respect to all objectives. In order to identify such solutions, the concept of Pareto dominance has been used. It offers a way to compare solutions and to classify them into non-dominated and dominated solutions. A solution dominates another one if it is at least as good as the other in all objectives and it is strictly better than the other in at least one objective. Although a great deal of effort has been devoted to solve multi-objective optimization problems, the problem is still open and the related issues still attract the interest of researchers. Most of the proposed approaches make use of metaheuristics. Pareto based metaheuristics for MOPs aim to introduce Pareto dominance into nature inspired algorithms. Population based metaheuristics such as Genetic Algorithms (GAs) and Particle Swarm Optimization (PSO) are particularly attractive to MOPs as they provide multiple non-dominated solutions in a single run [31][3].

Evolutionary Algorithms (EAs) have been largely investigated as stochastic search methods for handling multiple objectives. These methods are found to be well suited to solving a large number of MOPs. Moreover, EAs show the ability to cope with complex problems regardless of the shape of the Pareto surface [9][31].

However, EAs for MOPs share a common drawback, which is the overhead of the setting of many genetic parameters such as mutation, crossover, and selection [67]. That is why the PSO algorithm, a stochastic optimization technique inspired by social behaviour of bird flocking or fish schooling, has become an attractive alternative to solve MOPs due



to its simplicity, easy implementation, and less tunable parameters when compared with EAs [67][96]. Multi-Objective Particle Swarm Optimization (MOPSO) algorithms have been shown to be highly competitive with multi-objective genetic algorithms [92].

Recently, quantum computing principles have been introduced into EAs and PSO. While some of them focus on quantum representation of individuals [48], others suggest the use of a quantum behaviour defining a new philosophy for exploring the search space [9]. This latter was the basic idea behind the Quantum Behaved Particle Swarm Optimization (QPSO) algorithm, a quantum variant of PSO algorithm introduced by Sun et al. [105]. QPSO has been shown as a promising algorithm for many single objective optimization problems. Like PSO, QPSO is characterized by its simplicity and easy implementation. Besides, it has a better search ability and fewer parameters to adjust when compared against PSO [37]. Actually, only one tunable parameter is required. With the new search philosophy it suggests, QPSO could improve the convergence capability for global optimization [106].

The original PSO with related improvements or variants has been successfully applied to a large number of applications [93][111]. QPSO can be viewed as the result of introducing quantum mechanics principles into the traditional PSO. In this regard, QPSO proposes a new philosophy to explore the search space through the use of a quantum behavior while maintaining the spirit of the traditional PSO (i.e., the use of a swarm of particles that behaves according to local and global influence). Since its inception, QPSO has been subject to some improvements and has been shown to offer good performance when applied to a variety of problems [12][55][70]. Furthermore, QPSO presents some interesting characteristics [37][104] that may impact positively the performance of the search in a multidimensional space with multiple objectives. Hence, it is worth investigating how

it impacts a MOP algorithm’s behavior. These characteristics are basically summarized in the following points:

- The QPSO model introduces the use of the mean best position (*mbest*) which is considered as an improvement of the algorithm. In classical PSO, each particle converges directly to the global best position independent of the other particles in the swarm. This could lead to a premature convergence problem. However, with the mean best position in QPSO, particles do not converge to the global best position on their own without considering the other particles. The particles with self best positions far from the global best position are called lagged particles. Through *mbest*, the particle swarm never neglects any lagged particle. As a consequence, self best positions and mean best positions do not converge quickly to the global best position until lagged particles are close to the global best position. This “wait mechanism” [37] makes the particles close to the global best position able to explore globally around the global best. It is worth investigating the extent to which this wait mechanism will impact the search in a multi-objective context.
- QPSO has been shown to have a better search capability when compared with PSO. This is due to the fact that the PSO dynamic is based on Newtonian laws where a particle flies along a specified trajectory. Sun et al. [105][106] argue that the behavior of a swarm of birds or fish is much more complex than to be depicted by the basic Newtonian laws and suggest that quantum mechanics could be more suitable. Depicting the behavior by a quantum based model offers a new search philosophy that leads to better global optimality [37].
- Unlike PSO, QPSO has one tunable parameter to adjust which is the Contraction Expansion (CE) parameter  $\beta$ . While PSO algorithm has more parameters to set such as the inertia weight, the cognitive parameter which is related to the influence of

the self best performance, and the social parameter which is related to the influence of the global best performance. [37].

Based on the above characteristics of QPSO and the promising results reported in [105][106] comparing QPSO to PSO in the single objective domain, it is natural to ask whether it would be possible to extend its use to tackle MOPs in order to achieve better Pareto fronts in terms of convergence and diversity. Such an extension is not yet well investigated in the literature, to our knowledge. We found one published paper that used QPSO in a multi-objective context, by Omkar et al. [78] and deals with a specific application related to laminated composite structures. In this approach, the authors incorporated the vector evaluated technique within QPSO taking inspiration from Vector Evaluated PSO (VEPSO) [81].

### 1.3 Research Questions

In its current form, QPSO cannot be used in a straightforward way for MOPs. The question to be asked at the first glimpse is how can QPSO be extended to a multi-objective context to achieve a better balance between exploration and exploitation of the search process which would help to approach the targeted Pareto Front? In this work four important research questions are to be addressed:

- First, how to derive a local attractor of a given particle to compute its new position which in turn requires a selection strategy to select a leader for that particle and to decide about the policy for updating its self-best performance at each new position?
- Second, how to update the set of non-dominated solutions, which requires a decision about the archiving method?
- Third, how to handle multi-objective constrained problems using QPSO?

- Fourth, what is the potential of multi-objective quantum behaved particle swarm optimization when applied to a real-world problem?

## 1.4 Contributions

The main contributions of this thesis are:

- The development of a general framework of multi-objective quantum behaved particle swarm optimization, in which the global leader of each particle is selected by the proposed hybrid selection strategy.
- The development of a mechanism to handle the archive size, which we call the Redundancy Removal archiving method.
- The development of a constraint handling strategy for solving multi-objective constrained problems with QPSO. The design incorporates the infeasible solutions when computing the local attractors of particles and adopts a policy that achieves a balance between searching in infeasible and feasible regions.
- A novel application of the framework for solving the cluster analysis problem, which we call the MOQPSO-Clust.

## 1.5 Publications

- Heyam Al-Baity, Souham Meshoul, and Ata Kaban. On Extending Quantum Behaved Particle Swarm Optimization to Multi-objective Context. In Proceedings of the IEEE World Congress on Computational Intelligence (IEEE CEC 2012), pp. 1-8, 2012.
- Heyam Al-Baity, Souham Meshoul, and Ata Kaban. Constrained Multi-Objective Optimization using a Quantum Behaved Particle Swarm. The International Confe-

rence on Neural Information Processing (ICONIP 2012), Part III, LNCS 7665, pp. 456-464. Springer-Verlag Berlin Heidelberg, 2012.

- Heyam Al-Baity, Souham Meshoul, Ata Kaban and Lilac Alsafadi. Quantum Behaved Particle Swarm Optimization for Data Clustering with Multiple Objectives. IEEE Sixth International Conference on Soft Computing and Pattern Recognition, (IEEE SOCPAR), pp. 215-220, 2014.
- Heyam Al-Baity, Souham Meshoul, and Ata Kaban. Swarm Based Multi-Objective Optimization with Quantum Behaved Particles. International Journal of Bio-Inspired Computation (IJBIC), submitted, 2014.

## 1.6 Thesis Outline

This thesis consists of 8 chapters that are organized as follows:

Chapter 2 lays out the essential background and the basic concepts of multi-objective optimization problems including the problem formulation and the definition of dominance and Pareto optimality followed by an exposition of the differences between single and multi-objective optimization. We then give an introduction to swarm based metaheuristics followed by a description of particle swarm optimization and the paradigm shift from PSO to QPSO. At the end of this chapter, we provide a review of the most important approaches developed so far in the literature to solve multi-objective optimization problems and multi-objective particle swarm optimization problems.

Chapter 3 presents a new framework of Multi-Objective Quantum behaved Particle Swarm Optimization (MOQPSO). The chapter starts by describing the extension of QPSO to handle multiple objectives. This includes the definition of the proposed hybrid

leader selection strategy that selects the global leader for each particle and the policy we follow to maintain the set of non-dominated solutions throughout the search process. A comparative study is presented at the end of this chapter on the performance of MOQP-SO for unconstrained test problems against some of the state-of-the-art multi-objective evolutionary algorithms.

In Chapter 4, we propose the use of MOQPSO to solve constrained multi-objective optimization problems. A new constraint handling strategy is proposed which deals with both feasible and infeasible solutions when computing the local attractors of particles. The aim of this strategy is to maintain a balance between searching in infeasible and feasible regions and to obtain better results by incorporating infeasible solutions in the search process. This new strategy is then tested and compared with the death penalty constraint handling strategy. Discussion of results is presented at the end of the chapter.

Chapter 5 provides an extensive investigation of the potential of MOQPSO under different leader selection strategies on unconstrained and constrained test problems. In addition, a comparison between MOQPSO and MOPSO is performed. A discussion of results and conclusions derived from the experiments conducted are presented at the end of the chapter.

Chapter 6 presents a thorough empirical study of the influence of different archiving methods on the performance of MOQPSO on constrained and unconstrained test problems. A discussion of results and conclusions drawn from the experiments performed are presented at the end of the chapter.

Chapter 7 provides an application of the proposed MOQPSO framework in cluster

analysis problems. The search process is carried out over the space of cluster centroids with the aim to find out partitions that optimize two objectives simultaneously, namely compactness and connectivity. The proposed framework has been tested using both synthetic and real data sets and compared to the state-of-the-art methods.

In Chapter 8, we summarize the main contributions of the thesis, give conclusions of the proposed work, and state the possible future work directions.

### Background and Review of Related Work

---

The work in this thesis is at the intersection of two broad fields, namely optimization and nature inspired computing. Concerning the former, we are particularly interested in multi-objective optimization while in the latter our focus is on quantum behaved particle swarm optimization. Therefore, in this chapter we present the basic concepts and background knowledge necessary to follow the core of the present work. We also provide a review of some of the most relevant research approaches that have appeared in the literature related to multi-objective optimization in general and multi-objective particle swarm optimization in particular.

## 2.1 Basic Concepts of Multi-objective Optimization

### 2.1.1 Optimization Problems

Many real-world decision making problems require selecting the best element from a set of alternatives with regard to some criteria which is typically an optimization process. Indeed, optimization lies at the heart of many tasks in different domains ranging from science to industry, commerce, and finance, to name just a few. For example, in business



intelligence, optimization is involved in recommending near optimal decisions. Basically, optimization is the process of finding the best solution among all possible solutions of a problem without violating any of its constraints. More formally, it is a mathematical discipline that searches in the feasible region for a solution with the minimum or maximum value of the objective function. A potential solution of an optimization problem is usually defined as a decision vector composed of decision variables. A solution is feasible if it satisfies the problem's constraints. The set of all feasible solutions form the feasible region in the decision space [19][31].

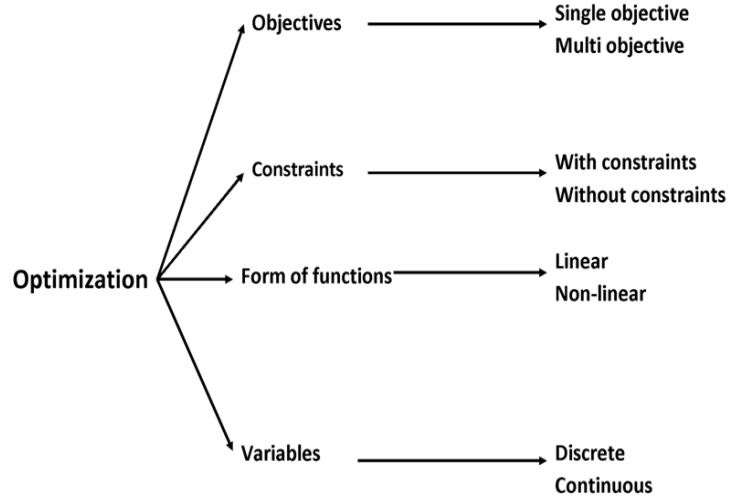
An optimization problem can be of several types:

- Constrained or unconstrained depending on whether it is subject to functional constraints or not.
- Single-objective or multi-objective depending on the number of objectives to be optimized.
- Linear or non-linear depending on the form of the objective and constraint functions.
- Continuous or discrete depending on whether decision variables are continuous or discrete.

These types are illustrated in Figure 2.1.

### 2.1.2 Single Objective Optimization Problems

As the name suggests, in a Single-objective Optimization Problem (SOP), the task is to search for values that optimize a single objective function [31]. For instance, buying a mobile phone with minimum cost is considered as a single optimization problem. The objective to be optimized in this case is minimizing the cost. A single optimization problem can be formulated as follows:



Figur 2.1: Types of optimization problems

A D-dimensional SOP consists of finding the decision vector  $\vec{x}^* = (x_1^*, x_2^*, \dots, x_D^*)^T$  that optimizes (minimizes or maximizes) an objective function  $F(\vec{x})$  subject to:

- Inequality and equality constraints that delimit the feasible region containing potential solutions

$$g_j(\vec{x}) < 0 \quad j = 1, 2, \dots, J$$

$$h_k(\vec{x}) = 0 \quad k = 1, 2, \dots, K$$

- Domain constraints are specified by the lower bound  $x_i^{(L)}$  and upper bound  $x_i^{(U)}$  of each decision variable  $x_i$ . They delimit a subspace of the D-dimensional space  $R^D$  called the search space which includes the feasible region [15].

$$x_i^{(L)} \leq x_i \leq x_i^{(U)} \quad i = 1, 2, \dots, D$$

A plethora of methods and techniques have been proposed to solve SOPs. A comprehensive review of these methods is beyond the scope of our work. Interested reader can refer to [112] for more details. Briefly, techniques devoted to solve SOPs can be broadly classified into two main categories namely deterministic and stochastic techniques - depending on the way exploration of the search space is performed.

Deterministic techniques find optimal solutions however they cannot be applied to all types of optimization problems. This class of methods includes hill climbing, mathematical programming techniques and enumerative methods among others.

Hill climbing is a local search algorithm that is popular for its simplicity and its greedy search nature. It is sensitive to initial starting point and gets stuck into local optima. As examples of methods from mathematical programming field, we can cite those numerical methods that are based on the gradient of the objective function. They iteratively refine a solution by moving in the objective space in the direction of the local gradient. However, they need to have differentiable objective functions, with their gradient available - which is often not the case in practice especially in black-box settings where only function evaluations are available, not their analytical forms. Another disadvantage is that gradient methods only guarantee to reach a local optimum.

Other mathematical optimization methods include the simplex method proposed by Dantzig [30] to solve linear programming, and have been also extended to deal with non-linear programming problems such as quadratic programming problems. However, it is

limited to linear and quadratic programming.

Enumerative methods, as their name suggests, seek for the optimal solution by performing evaluation of each and every feasible solution of a finite or discretized infinite search space. Examples of such methods include Branch and Bound and Dynamic programming. Obviously, these methods perform well in case of small size search spaces and cannot be applied to solve problems of even moderate size and complexity. On the other hand, stochastic methods have been developed to deal with large size search spaces with the aim to seek for good quality solutions in a reasonable time. They propose a stochastic exploration of the search space. Meta-heuristics like genetic algorithms, differential evolution, particle swarm optimization and others have been largely used for this purpose.

### **2.1.3 Multi Objective Optimization Problems**

Most real-world optimization problems involve multiple conflicting objectives that should be optimized (minimized or maximized) simultaneously. This simultaneous optimization could provide better quality solutions and better insights to the problem which in turn will help making better decisions. Such problems are known as Multi-objective Optimization Problems (MOPs). In MOP, there is no longer a single optimum solution, but rather a set of trade-off solutions known as the Pareto optimal set or Pareto optimal solutions from which the decision maker can select one according to his/her preference. These solutions are optimal in a sense that no other solution is superior when all objectives are considered [31][3].

For example, consider the problem of determining the most efficient choice for purchasing a mobile phone. Assume we use two criteria to be optimized: the width of the mobile screen (to be maximized) and the price or the cost of the mobile (to be minimized). Due

to the conflicting relationship between these two objectives, many trade-off solutions can be found. For instance, the buyer can choose one of the trade off solutions (A, B, C) that are shown in Figure 2.2. If the buyer is concerned about the cost, then he/she can choose solution A. If the buyer is interested in a wide screen mobile, then he/she can choose solution C.

## MOP Formulation

Three basic elements define any MOP, namely: a set of decision variables, a set of objectives and a set of constraints. Basically, a MOP consists of exploring the search space of the decision variables in order to find the vector of variables that optimizes the set of objectives while satisfying the set of constraints. More formally, a MOP is defined as follows [31][15]:

A D-dimensional multi-objective optimization problem consists of finding a decision vector  $\vec{x}^* = (x_1^*, x_2^*, \dots, x_D^*)^T$  that optimizes (minimizes or maximizes) a vector of M objectives  $\vec{F}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \dots, f_M(\vec{x}))$  subject to:

- Inequality and equality constraints that delimit the feasible region containing potential solutions

$$g_j(\vec{x}) < 0 \quad j = 1, 2, \dots, J$$

$$h_k(\vec{x}) = 0 \quad k = 1, 2, \dots, K$$

- Domain constraints are specified by the lower bound  $x_i^{(L)}$  and upper bound  $x_i^{(U)}$  of each decision variable  $x_i$ . They delimit a subspace of the D-dimensional space  $R^D$  called the search space which includes the feasible region [15].

$$x_i^{(L)} \leq x_i \leq x_i^{(U)} \quad i = 1, 2, \dots, D$$

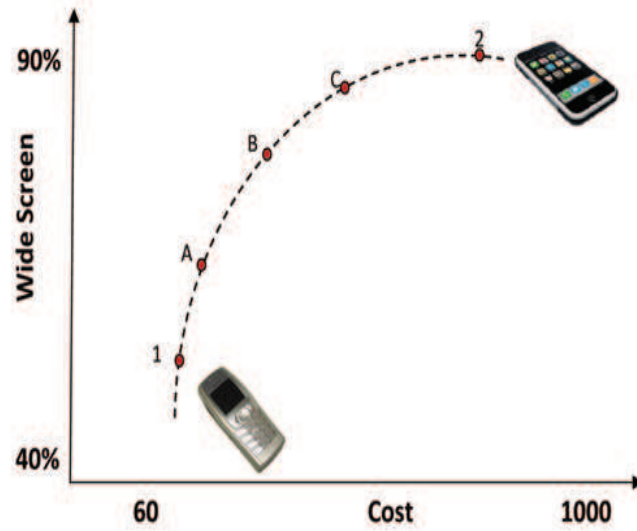


Figure 2.2: Simple example of a MOP inspired by [31]

## Dominance and Pareto Optimality

The notion of optimality does not apply directly to the multi-objective context as in the single objective optimization domain. In the context of single objective optimization problems, the solutions are easily compared against each other according to their objective values. The question is how to handle the comparison among solutions in the multi-objective context? Pareto Dominance is the concept used for this purpose. For instance, we say that solution  $x_1$  dominates solution  $x_2$  if it is better than  $x_2$  in at least one objective and not worse than  $x_2$  in all other objectives. In this case  $x_1$  is better than  $x_2$  and we call  $x_1$  a non-dominated solution. Therefore, a non-dominated solution is the one that

dominates all other potential solutions to the problem. Moreover, if  $x_1$  does not dominate  $x_2$  and vice versa, then both  $x_1$  and  $x_2$  are considered as non-dominated solutions [31].

- **Pareto Dominance**

Definition [22]: A vector  $u = (u_1, \dots, u_k)$  is said to dominate vector  $v = (v_1, \dots, v_k)$  denoted by  $(u \preceq v)$  if and only if (in the minimization case):

$$\forall i \in \{1, 2, \dots, k\}, u_i \leq v_i \wedge \exists i \in \{1, 2, \dots, k\} : u_i < v_i$$

Therefore, the best solutions with respect to the Pareto dominance relation constitute the Pareto optimal set and are called non-dominated solutions.

- **Pareto Optimal Set**

Definition [3][22]: A solution is said to be Pareto optimal if and only if it is not dominated by any other solution in the search space  $\Omega$ . The set of all Pareto optimal solutions is called Pareto optimal set  $P^*$  and is defined as:

$$P^* = \{\vec{x} \in \Omega | \neg \exists \vec{x}' \in \Omega \vec{f}(\vec{x}') \preceq \vec{f}(\vec{x})\}$$

- **Pareto Front**

Definition [22]: Each solution vector in the Pareto optimal set corresponds to a vector of the related objectives values. Therefore, a Pareto front represents the set of Pareto optimal solutions in the objective space [22][31]. The Pareto front is defined as:

Given a MOP  $\vec{F}(x)$  and Pareto optimal set  $P^*$ , the Pareto front  $PF^*$  is defined as :

$$PF^* = \{\vec{u} = (f_1(x), \dots, f_k(x)) | x \in P^*\}$$

Figure 2.3 shows an example of a Pareto front of a MOP with two objective functions and some other solutions dominated by the Pareto front solutions.

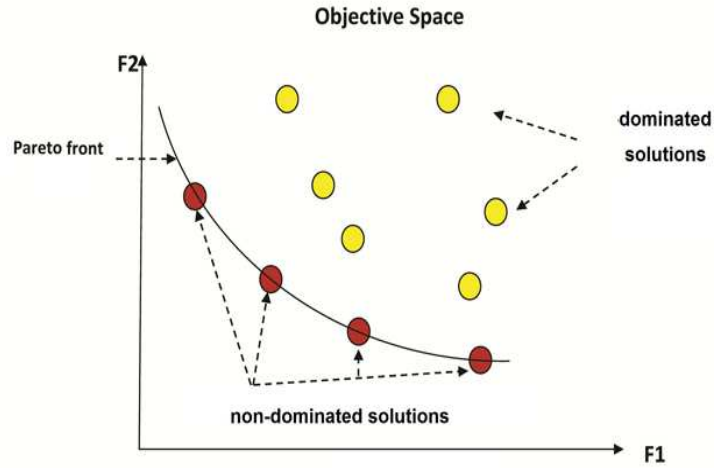


Figure 2.3: Example of a Pareto front and dominated solutions in the objective space for a minimization problem. Taken from [31].

## SOP versus MOP

Figure 2.4 illustrates the difference between the single objective and the multi-objective optimization problems. We can see from Figure 2.4 (a) that we need to find the best solution which is shown as the global optimum point in the figure. Figure 2.4 (b) shows the curve that represents the set of optimal solutions we have to find in the case of MOP.



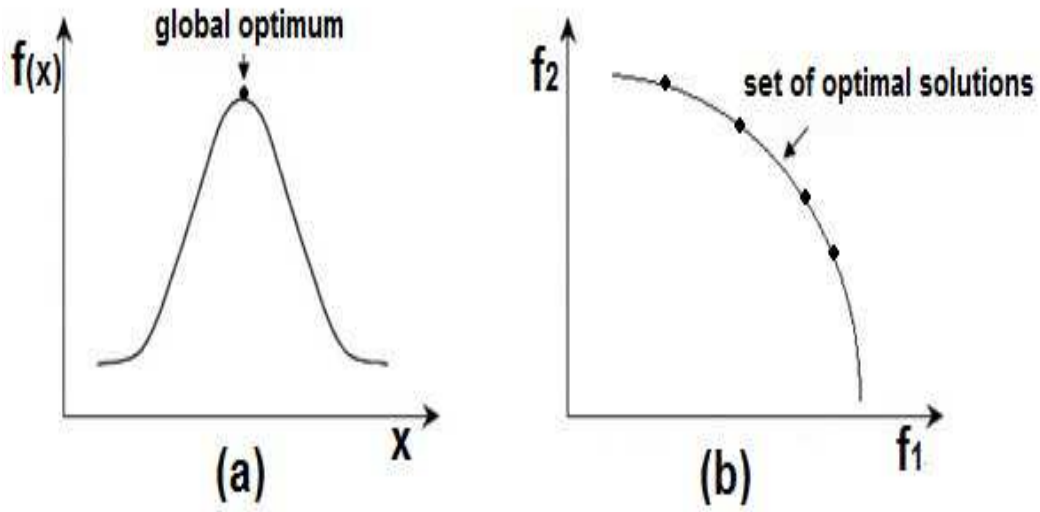


Figure 2.4: Difference between SOP and MOP inspired by [31].

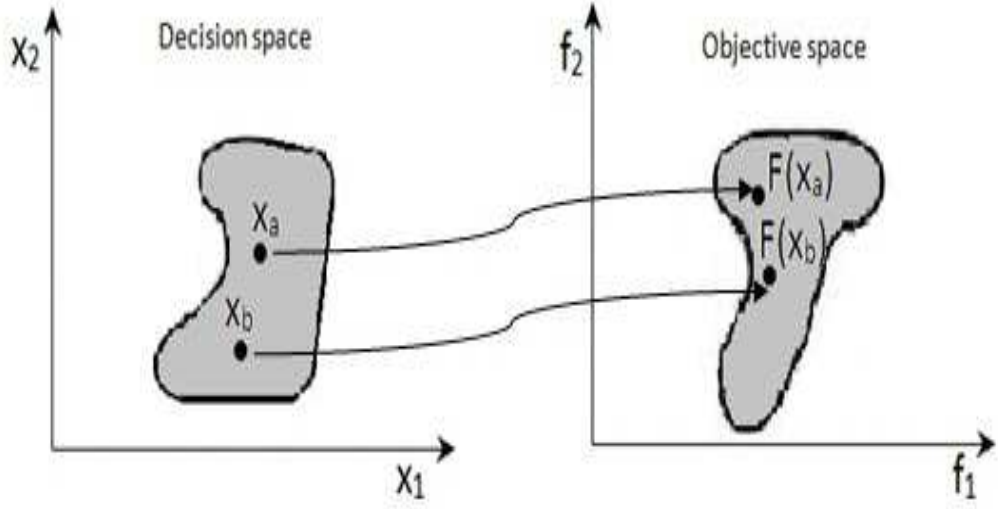
With SOP, we deal with only one search space which is the decision space. While in MOP, we deal with two search spaces, namely the decision space and the objective space. The objective space is defined as the space in which the objective vector belongs. The decision space is the feasible search space of the problem in which the decision variables belong [31]. Table 2.1 summarizes the differences between single objective and multi-objective optimization problems.

SOP	MOP
One objective function	Multiple objective functions
One search space (decision search space)	Two search spaces(decision and objective search spaces)
Interested in one optimal solution	Interested in a set of optimal solutions
Requires search only	Requires search and decision making
Optimality is related to objective fitness value	Optimality is related to dominance concept

Table 2.1: SOP vs. MOP

The main difference between a single objective and a multi-objective optimization pro-

blem is that in the case of multi-objective optimization, a multi-dimensional space called the objective space is constituted by the objective functions. Therefore, each solution in the decision space maps to a point in the objective space. Figure 2.5 shows the mapping process from decision space to objective space.



Figur 2.5: Mapping between decision space and objective space taken from [31].

The challenge in a multi-objective optimization problem is to find the Pareto optimal solutions that are as close as possible to the true Pareto optimal front and to maintain diversity in this developed Pareto optimal set [31].

Multi-objective optimization problems have been solved in different ways by several methods as will be seen in the review of related work given in section 2.3. Some of these methods produce only one solution at each run. Therefore to obtain a set of compromise solutions, several runs are required. This issue has raised interest toward population based metaheuristics as they can produce several solutions at each run and they exhibit good

search capabilities in the search space [31][25]. In our study, we are interested in swarm based metaheuristics for which a presentation is given in section 2.2.

#### 2.1.4 Computational Complexity Considerations

Complexity analysis aims at quantifying the amount of time and space, among others, required for solving problems. It is usually required for problems where the space of possible solutions is very large. It allows comparing different algorithms designed to solve a problem. For instance, it helps to know whether an algorithm A is better than another algorithm B in terms of time or space complexity, if it is optimal or if it cannot be used. Time complexity can be recorded according to three situations or scenarios namely the worst case, the best case, and on average. Different degrees of complexity can be defined ranging from the lowest namely logarithmic complexity going through polynomial complexity to the highest namely exponential complexity.

The branch of Theoretical Computer Science that deals with complexity is Computational Complexity Theory (CCT). CCT makes use of mathematical models to study the inherent difficulty of solving a computational problem. It aims to determine the practical limits on what computers can and cannot do. Basically, it focuses on decision problems, that is problems that verify whether a given input satisfies a certain property and give a YES/NO answer. For the other classes of problems, a decision version can be derived for a given problem. For example, for a minimization problem where we seek for the solution that optimizes a given objective  $f$ , a decision version of this problem could be the following: Is there a solution  $S$  that satisfies  $f(S) < V$ , such that  $V$  is a given objective value?

A problem may be regarded as easy or difficult to solve. A problem is classed as a

difficult problem if it requires significant resources, such as time and storage. Basically, we are interested in how algorithms scale with an increase in the input size.

The resource required to solve a problem is calculated as a function of the size of the problem instance. For example, the worst-case time complexity  $T(n)$  is defined to be the maximum time taken over all inputs of size  $n$ . In CCT, different classes of complexity are defined to classify problems according to the complexity of the algorithms used to solve them [110]. The well-known classes of complexity encompass the class P and the class NP. The class P consists of problems that are solvable in polynomial time (where P stands for Polynomial time) that is, there is some polynomial  $p$  such that the algorithm runs in time at most  $p(n)$  on inputs of length  $n$ . Thus, the P class includes those problems that are considered easy. The class NP comprises the problems that are solvable in polynomial time by a non-deterministic Turing machine. The term NP stands for Non-deterministic Polynomial [110][36].

The reduction concept is used to define and relate the different complexity classes. A reduction can be regarded as the transformation of one problem into another one. A problem A can be reduced to a problem B if A can be solved using an algorithm for B. A is reduced to B means that the problem A is no more difficult than the problem B. For example computing a power of a number can be reduced to multiplication. Many types of reductions exist in the literature. The most used one is the polynomial time reduction [110] where the reduction process is a polynomial time task.

The concept of reduction helps defining NP-hard problems and NP-Complete problems. A problem is hard for a class of problems if every problem in this class can be reduced to it which means no problem in this class is harder than it. Therefore, the set

of problems that are hard for NP is the set of NP-hard problems [110]. More simply, a problem is NP-hard when every problem in class NP can be reduced in polynomial time to it. Problems in NP-hard do not have to be elements of NP, as they may not even be decidable problems. A problem is said to be complete for a given class if it is a problem of this class and if it is hard for this class. Therefore, the class of NP-complete problems contains the most difficult problems in NP. Each element of NP-complete has to be an element of NP [36][110].

### 2.1.5 Why Metaheuristics for Continuous MOPs

In our work, we tackle multi-objective continuous optimization problems. In these problems, the task is to find the values of  $D$  decision variables  $x_i$ ,  $i = 1..D$  that are defined over a continuous range of values (usually real numbers) and that optimize a vector of objective functions. The general MOP formulation given in section 2.1.3 can be rewritten in the context of continuous problems as follows:

A D-dimensional multi-objective optimization problem consists of finding a decision vector  $\vec{x}^* = (x_1^*, x_2^*, \dots, x_D^*)^T \in R^D$  that optimizes (minimizes or maximizes) a vector of M objectives  $\vec{F}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \dots, f_M(\vec{x}))$  subject to:

- Inequality and equality constraints that delimit the feasible region containing potential solutions

$$g_j(\vec{x}) < 0 \quad j = 1, 2, \dots, J$$

$$h_k(\vec{x}) = 0 \quad k = 1, 2, \dots, K$$

- Domain constraints are specified by the lower bound  $x_i^{(L)}$  and upper bound  $x_i^{(U)}$  of each decision variable  $x_i$ . They delimit a subspace of the D-dimensional space  $R^D$  called the search space which includes the feasible region [15].

$$x_i^{(L)} \leq x_i \leq x_i^{(U)} \quad i = 1, 2, \dots, D$$

Obviously, the search space is exponential in the problem dimension. Suppose that all decision variables are defined over the smallest range  $[B_l..B_u]$  (given  $B_l$  the lower bound and  $B_u$  the upper bound), then the number of possible solutions is  $B^D$  where  $B$  is the number of possible values within the range  $[B_l..B_u]$  obtained using a step increment. Although the range  $[B_l..B_u]$  is bounded, the number of possible real values in the D-dimensional search space  $[B_l..B_u]^D$ , using a discretization with a fixed step increment, grows exponentially with D. In general, such problem is NP-hard and it is very difficult if not impossible to come up with an algorithm that produces an optimal or close to optimal solution within a time bound that is polynomial in the problem dimension. Furthermore, the number (multiple objectives) and type of objectives and constraints (linearity and convexity) make the problem even more difficult to solve using the standard methods from the field of mathematical programming such as non linear programming. Therefore, a global search heuristic method is needed to find approximate solutions within a reasonable amount of time.

## 2.2 Introduction to Swarm Based Metaheuristics

With the complex real-world optimization problems, conventional optimization methods become inefficient in that it takes prohibitively long computational time to get exact solutions. Therefore, recently, general purpose stochastic algorithms have emerged for finding approximate solutions to such hard optimization problems (i.e., problems that involve very large search spaces as explained in section 2.1.4). One such class of algorithms is referred to as metaheuristic algorithms. These algorithms are often inspired by mechanisms taken from nature. Therefore, they are referred to as nature inspired algorithms. These methods have become the focus of research due to their efficiency, flexibility, and broad

applicability [43].

Swarm intelligence based algorithms (SI) are a subset of nature inspired metaheuristic algorithms. These represent a new computation technique that is inspired by the collective intelligence behavior of a group of social insects such as bees, ants, or wasps [111]. There is no clear definition for swarm intelligence but it can be described as “a collective behavior of decentralized, self organized systems” [111]. There are many kinds of living organisms in nature that exhibit the social behaviors and self-organization systems such as bird flocking, fish schooling, and ant colonies. The individuals of these various social animals aggregate in groups called swarms. These individuals interact with each other and collaborate in order to accomplish a certain task like finding a food source. In the case of bird flocking, the flock will move towards the food area as a single unit with no previous plan nor a centralized leader. The flock movement is accomplished based on some simple rules of the individual birds which allow them to coordinate their movements with their mates in the flock. Each bird contributes to its flock by sharing its experience and gets benefit from the flock by taking their experiences. In other words, all the flock birds share their experiences with each other to reach their goal. This kind of sharing leads to a global behavior without any supervision [64].

The swarm intelligence field has gained wide popularity in the past decades and is becoming a powerful Artificial intelligence tool for solving difficult optimization problems. Particle swarm optimization, ant colony optimization, bee colony optimization and wasp colony optimization are some popular swarm intelligence algorithms [111][64].

### 2.2.1 Particle Swarm Optimization

Particle Swarm Optimization (PSO) techniques are a form of swarm intelligence based methods that are developed based on the social behavior of bird flocking or fish schooling. These are population-based stochastic optimization methods developed by Dr. Eberhart and Dr. Kennedy in 1995. In particle swarm optimization, the collection of birds or fish constitute the swarm. The birds or fish are represented by points called particles. The PSO algorithm attempts to find the optimal solution through moving the particles toward better regions in the search space influenced by the improvements obtained by the other particles in the swarm. PSO follows the principle of bird flocking or fish schooling in that they start searching for a food source without any idea about its location but with the interaction and sharing mechanism they follow, they can finally reach the best location of the food source [63]. Particle swarm optimization is very effective for continuous optimization problems. It is similar to evolutionary computation in that [111][93][96]:

1. Both initialize the population with random solutions.
2. Both search for the global optimum over generations.
3. Reproduction of new generations is based on previous ones.
4. Both use the fitness function to evaluate a potential solution.
5. Both use randomized techniques to update the population.

However, PSO differs from evolutionary computation in that:

1. PSO does not have evolutionary operators such as crossover and mutation. The potential solutions in PSO update their positions based on their own experience and the experience of other particles in the swarm.



2. There is no selection mechanism in PSO. All particles will survive, unlike in genetic algorithms, where the low fitness individuals will die.
3. PSO uses an internal velocity to direct the particles in their movement within the search space. This can be viewed as a directional mutation. By contrast, in evolutionary computation, the mutation operator sets the individuals into any direction.

PSO consists of a collection of particles which represent the problem search space, where each particle represents a candidate solution. In PSO, the swarm is initialized with random positions. The potential solutions (particles) will move through the search space toward better areas where the optimal solutions may reside by following the current best particles. Each particle in the swarm holds three vectors. These are:

- The current position (the candidate solution to the problem) denoted as  $X_i = (x_{i1}, x_{i2}, \dots, x_{iD})$ .
- A memory of best position found by the particle so far called the personal best position or self best (*sbest*) denoted as  $sbest_i = (sbest_{i1}, sbest_{i2}, \dots, sbest_{iD})$
- The velocity denoted as  $V_i = (v_{i1}, v_{i2}, \dots, v_{iD})$ , where D is the dimension of the search space.

In PSO, a candidate solution is encoded in terms of a position of the particle. During each time step, each particle adjusts its position toward the current optimum particles according to the best position found by itself (*sbest*) and the best position achieved by the whole swarm called the global leader or global best position (*gbest*). The global best position acts as the guide to the swarm in its search, i.e., all the swarm particles will follow the *gbest* during the search process [111][83].

## The Basic Particle Swarm Optimization

In a D-dimensional problem space, each particle is defined by a vector of positions representing a candidate solution and a vector of velocities that defines the amount of change in the position of a particle. When moving in the search space, a particle is influenced by its self-best position  $sbest_i = (sbest_{i1}, sbest_{i2}, \dots, sbest_{iD})$  and the best position among all particles in the swarm  $gbest = (gbest_1, gbest_2, \dots, gbest_D)$ . The velocity and the position of a particle  $i$  at time  $(t + 1)$  are updated according to the following equations [97][98]:

$$V_{ij}^{t+1} = W.V_{ij}^t + C1r_{1j}^t(sbest_{ij}^t - x_{ij}^t) + C2r_{2j}^t(gbest_j^t - x_{ij}^t) \quad (2.1)$$

$$x_{ij}^{t+1} = x_{ij}^t + V_{ij}^{t+1} \quad (2.2)$$

where

- $W$  is the inertia weight that plays a role in maintaining a good balance between the global and local search ability of PSO. larger values of  $W$  will promote global exploration of the search space. However, smaller values of  $W$  will favor local exploitation. Therefore, setting the value of  $W$  is quite a difficult task as it has a great impact on the algorithm convergence.
- $V_{ij}^{(t)}$  is the velocity at time  $t$  for dimension  $j$  of particle  $i$ .
- $C1$  and  $C2$  are two positive acceleration constants that balance the influence of the particle's self best position and that of the swarm respectively for moving the particle towards the target.  $C1$  and  $C2$  are called the cognitive and social constants as they determine the weight of attraction to  $sbest$  and  $gbest$  respectively.

- $r_{1j}$  and  $r_{2j}$  are two random numbers within the range  $[0, 1]$  for dimension  $j$ , which are used to maintain the population diversity.
- $sbest_{ij}^{(t)}$  is the self best position recorded by the particle  $i$  so far for dimension  $j$ .
- $x_{ij}^{(t)}$  is the position for dimension  $j$  of particle  $i$  at time  $t$ .
- $gbest_j^{(t)}$  is the global best position of dimension  $j$  found by the swarm so far.

The update velocity in equation (2.1) consists of three terms. The first term is called habit or momentum. This term tends to keep the impact of the previous velocity when computing the current velocity by moving the particle in the same direction as previously. The second term is known as the cognitive part or self influence of a particle which pulls the particle towards its own best position. This term represents the local search ability of the particle. The third term which is known as cooperation or social influence will move the particle towards the global best position of the entire swarm and is related to the global search ability of the particle [97][98].

## Other Variants of PSO and Applications

In order to improve the premature convergence and the global optimization ability of the classical PSO, many changes have been proposed to PSO parameters such as the swarm size, the acceleration coefficients ( $C1$ ,  $C2$ ) and the inertia weight ( $W$ ). Moreover, the update equation of a particle's velocity is a key factor in PSO variants. An example of a PSO variant is the Discrete PSO, which is designed to solve discrete optimization problems. Another PSO variant is the Bare-bones PSO. This variant does not use the position or the velocity update equations. Instead, it uses a procedure that is similar to a parametric probability density function in order to update the particles positions [93][111].

PSO has been successfully applied to a broad range of application areas including neural network training, scheduling, forecasting, feature selection, telecommunications, data mining and many more [93][111]. A good review of the PSO metaheuristic can be found in [83][111].

## Advantages and limitations of PSO algorithm

The following are some advantages of PSO algorithm [11][93]:

1. Simple algorithm with easy implementation.
2. Few parameters that need to be adjusted compared with the genetic evolutionary algorithms.
3. Has been applied successfully in many application areas such as neural network and function optimization.
4. The velocity calculation is very simple.
5. Fast convergence to global optima.
6. Can take real numbers as particles.
7. Efficient computational cost.

However, the risk of PSO to be trapped in local optima is very high as the algorithm has the tendency to find the near optimal solution quickly rather than finding the optimal global one. And hence, all particles are grouped around this solution causing a premature convergence that could provide low quality solutions. As a result, the algorithm may suffer from lack of solution diversity as well [92].

## 2.2.2 From PSO to QPSO

The basic PSO algorithm suffers from the premature convergence problem because if the global best particle is trapped in a local minimum, all particles will quickly converge to the position of the global best particle found so far. Hence, the matter is how to let the search explore areas far from the local attractor of particles instead of focusing only on the neighbourhood of these local attractors. This issue has encouraged several attempts to further enhance the diversification or exploration ability of searching with PSO. Improved results have been achieved by introducing EA operators like mutation or hybridization of PSO with other metaheuristics [37][92].

In order to avoid the drawbacks of the PSO algorithm, some authors found other variants of PSO to improve its performance. Others like Sun et al. rewrote the PSO model by developing a new evolutionary equation that does not need the velocity vector. At the same time it follows the same principle of PSO in that we have a swarm of particles and each particle moves in the search space under the influence of its self best performance and the global best performance of the entire swarm. Encouraged by the fact that the social behavior of a swarm is too complicated to be depicted by classical mechanics, Sun et al. [105] merged the classical PSO algorithm with quantum mechanics resulting in the emergence of the Quantum behaved PSO algorithm termed as (QPSO).

QPSO changed and improved the search strategy of PSO by introducing the use of a new global point called the mean best position (*mbest*), which is the average of the self best positions of all particles. One of the disadvantages of PSO is that each particle converges to the global best position directly without waiting for the remaining particles in the swarm, and this could cause a premature convergence problem. However, with the *mbest* feature in QPSO, particles in the swarm do not converge to the global best position

directly without considering their mates. That is, particles that are close to the global best position will wait for the lagged particles (the particles that are far from the global best position) until they become closer to the global best position. This wait mechanism allows the particles close to the global best position to explore globally in the area around the global best position. As a result, QPSO is able to maintain a good balance between exploration and exploitation in the swarm, which decreases the risk of facing the premature convergence problem that is typically observed with PSO.

Sun et al. [105] found that the position of a particle includes two basic terms. The first term is nothing other than the attractor of the particle and the second term is related to the gap between the particle's current position and the mean best performance of the whole swarm. Hence, the particle position at the  $(t + 1)^{th}$  iteration is updated according to the following equations [105][104]:

$$x_{ij}^{t+1} = p_{ij}^t \pm \beta \cdot |mbest_j^t - x_{ij}^t| \cdot \ln(1/u_{ij}^t) \quad (2.3)$$

where:

- $p_{ij}^t$  is the local attractor and is evaluated by:

$$p_{ij}^t = \varphi_{ij}^t \cdot sbest_{ij}^t + (1 - \varphi_{ij}^t) \cdot gbest_j^t \quad (2.4)$$

with  $\varphi_{ij}^t = rand(0, 1)$

- $\beta$  is the Contraction Expansion coefficient (CE). It is the only tunable parameter of QPSO and has a significant impact on controlling the convergence speed of the algorithm [37].
- $mbest$  called the Mainstream Thought point or the mean best position. It is the

mean of *sbest* positions of all particles and is evaluated by:

$$mbest^t = \frac{1}{N} \sum_{i=1}^N sbest_i = (\frac{1}{N} \sum_{i=1}^N sbest_{i1}, \frac{1}{N} \sum_{i=1}^N sbest_{i2}, \dots, \frac{1}{N} \sum_{i=1}^N sbest_{iD}) \quad (2.5)$$

provided that N is the population size and D is the problem dimension of the search space.

- $u_{ij}^t$  that appears in equation (2.3) is a random number within the range [0,1].

Since its inception, many improved versions have been proposed to enhance its exploration and exploitation capabilities. A good review of QPSO with related improvements can be found in [37][104]. As described by the outlines of PSO and QPSO given in Algorithms 1 and 2, both have the same general dynamics. The difference between them lies in the equations that govern their dynamics.

---

**Algorithm 1** Pseudocode of PSO

---

- 1: Initialize swarm (position and velocity vectors)
  - 2: Initialize self best particles
  - 3: Locate global best particle (*gbest*)
  - 4: Initialize PSO parameters
  - 5: **while** (not termination-condition) **do**
  - 6:   **for** Each particle **do**
  - 7:     update position according to equation (2.2)
  - 8:     evaluate position
  - 9:     update self best particle (*sbest*)
  - 10:   **end for**
  - 11:   Update global best particle (*gbest*)
  - 12: **end while**
- 

Generally, when dealing with a global optimization problem, a crucial question that

---

**Algorithm 2** Pseudocode of QPSO

---

```
1: Initialize swarm (position vectors)
2: Initialize self best particles
3: Locate global best particle (gbest)
4: Initialize contraction-expansion parameter
5: while (not termination-condition) do
6:   Compute mbest position according to equation(2.5)
7:   for Each particle do
8:     update position according to equation (2.3)
9:     evaluate position
10:    update self best particle (sbest)
11:  end for
12:  Update global best particle (gbest)
13:  Decrease contraction-expansion parameter linearly
14: end while
```

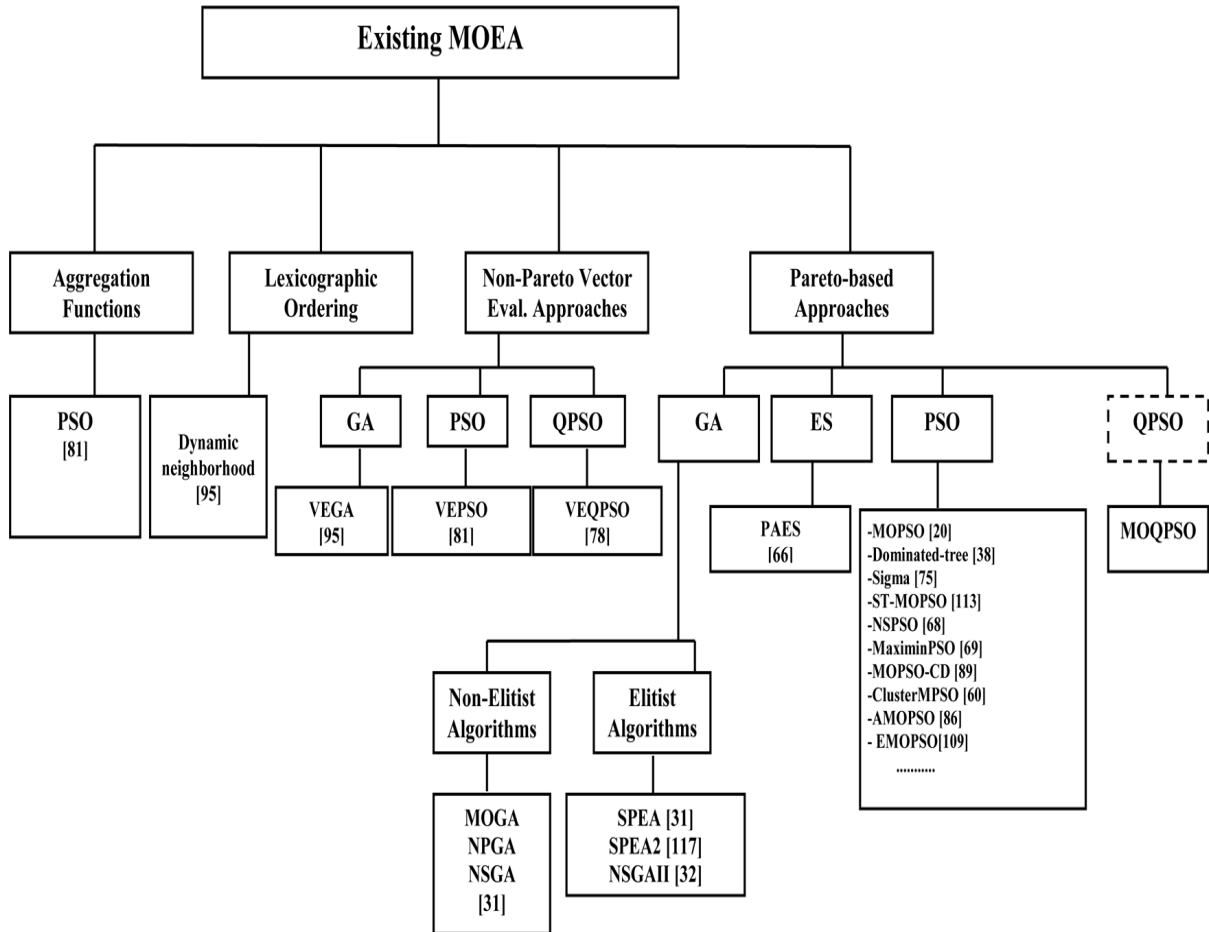
---

might arise is how to explore the search space in order to find good quality solutions. Therefore, the evolutionary computation algorithms can be classified according to the mechanism they adopt to move through the search space. In GAs, the global optimization process is performed through the genetic operators, selection, crossover, and mutation. The GA evolves through iterations by creating new subsequent populations from individuals in previous generations [108]. However, the optimization process in QPSO is performed using its evolutionary update equation. By this equation, the position of each particle of the swarm, which represents a potential solution to the problem, is updated under the influence of the self best performance located by the particle so far, the global best performance of the whole swarm so far, and the mean of the self best positions of all particles. It can be noticed that QPSO updates its population according to the social behaviour of the swarm by following the best particle. This mechanism facilitates the improvement of the particles positions through generations. In this way, the global best position is updated and improved through generations as well [105].



## 2.3 Review of the Past Related Work

A great deal of effort had been devoted to solving multi-objective optimization problems in the literature. A comprehensive review of related methods can be found in [23]. Existing Multi-Objective Evolutionary Algorithms (MOEA) fall into four classes as shown in Figure 2.6 where a summary of the most important methods is given. This review emphasizes the Pareto-based approaches and more specifically the PSO-based ones.



Figur 2.6: Taxonomy of multi-objective evolutionary algorithms

In the sequel we will describe the various evolutionary approaches to solving multi-objective optimization problems.

### 2.3.1 Weighted Sum Aggregation Function Based Approaches

This algorithm was proposed by Hajela and Lin [46]. It is the simplest method of the classical non-Pareto EA approaches that is based on plain aggregation. Each objective is assigned a weight value reflecting its importance such that the sum of all weights is one ( $\sum w_i = 1$ ). All weighted objective functions are summed into a single fitness value. In this way, the multi-objective functions are transformed into a single objective function  $\sum_{i=1}^k w_i f_i(x)$  where  $w_i$  are the weights and  $k$  is the number of the objective functions [31][120].

The shortcoming of this approach is the difficulty of specifying the weights for the objective functions which requires a prior knowledge of the objectives' importance [31]. In addition, the performance of this method is influenced by the Pareto front shape. For instance, the weighted sum method fails to find the Pareto optimal solution in concave regions [81]. Furthermore, the aggregated function generates only one solution in a single run and hence the trade-offs between the objectives cannot be evaluated easily [31]. Zitzler et al. in [120] compared four multi-objective EAs (among them VEGA and weighted aggregation function) on a multi-objective 0/1 knapsack problem with nine different parameter settings. The results showed that VEGA outperformed the weighted sum method. VEGA method will be described later in section 2.3.3.

### 2.3.2 Lexicographic Ordering Based Approaches

In this approach, a predefined preference (ordering) is specified among objective functions. Then each objective is optimized separately according to the assigned order of importance. The drawback of this approach is that it requires a prior knowledge of the objectives'

importance in order to set the ordering. Moreover, this approach is similar to the weighted sum method in that it finds only one single solution in each run [24][31].

Under the umbrella of this class of methods, Dozier et al. [35] proposed an algorithm that can be used for multi-objective path planning systems. This algorithm adopted the lexicographic preferences for selecting the candidate paths together with some complicated fuzzy set-based methods [73]. The results show that the algorithm was able to converge to optimal or near optimal paths. Also, the idea of lexicographic ordering has been adopted by Hu and Eberhat [56] who proposed a dynamic neighbored PSO algorithm. In this algorithm, objective functions are ranked according to their importance. Then each objective is optimized separately following the order of importance. A PSO variant with dynamic neighborhoods was incorporated. This approach is suitable for two or three objective optimization problems because when the number of objectives grows, it becomes too difficult to establish a good ordering for the objectives and this will heavily affect the performance of the algorithm.

Generally, the classical methods for solving MOPs as the weighted sum and lexicographic ordering require a high computational cost due to the several runs that have to be performed to obtain a set of non-dominated solutions. Moreover, they require a prior problem knowledge such as suitable objective weights and importance of objectives. Furthermore, these algorithms face difficulties in solving non-convex problems [31][81].

### **2.3.3 Non-Pareto Vector Evaluated Approaches**

Three algorithms have been proposed for this approach. The Vector Evaluated Genetic Algorithm (VEGA), Vector Evaluated Particle Swarm Optimization (VEPSO), and Vector Evaluated Quantum Particle Swarm Optimism (VEQPSO).

- **Vector Evaluated Genetic Algorithms (VEGA)**

This algorithm, the first GA dealing with multi-objectives, was proposed by Schaffer [95] during the mid 1980s. It is an extended version of a single GA to handle multi-objective problem in a single run. VEGA divides the population into subpopulations according to the number of the objective function to be optimized. Each subpopulation is controlled by its own objective function. Then the algorithm performs a selection mechanism for each objective function independently in order to find the optimal solutions for each objective in each subpopulation. Crossover and mutation are applied on the selected solutions to create the next generation [24][31]. Although it is simple and easy to implement, VEGA tends to converge to one best solution of one objective function neglecting the other objectives, i.e., the concept of trade off between objectives is missing [31]. Moreover, VEGA is unable to preserve solutions with promising performance along the run [24]. Finally, VEGA is not well suited for problems with concave surfaces [40].

- **Vector Evaluated Particle Swarm Optimization (VEPSO)**

This algorithm was suggested by Parsopoulos and Vrahatis [81]. In VEPSO, the idea of VEGA [95] was adopted in the PSO algorithm. Two swarms were used, one for each objective function. In addition, the weighted aggregation approach was also adopted. Experiments were performed to solve five benchmark problems. The results showed that the conventional weighted sum was able to give good results when the test functions were convex. In the concave case, the weighted sum could not obtain the Pareto optimal solutions. Whereas, VEPSO was able to converge near the Pareto front. However, it is only designed for two objective problems as it tends to divide the swarm into subswarms based to the number of objectives to be optimized. When dealing with a large number of objectives, the subswarm size becomes small and this will cause a deterioration in the diversity of the obtained

solutions. As a consequence, the algorithm will not be able to converge to the entire Pareto front.

- **Vector Evaluated Quantum Particle Swarm Optimization (VEQPSO)**

Few research on extending QPSO to multi-objective optimization have been reported in the literature. A paper proposed by Omkar et al. [78] was devoted to the design of a genetic framework for multi-objective optimization for laminated composite structures. In this approach, a multi-objective optimization algorithm that is based on vector evaluated quantum PSO (VEQPSO) is presented. The authors of the paper incorporated a vector evaluated technique within QPSO taking inspiration from Vector Evaluated PSO (VEPSO) [81]. The performance of QPSO is compared against PSO. QPSO shows slower convergence to the Pareto front than the PSO. However, QPSO has better global search capability than PSO.

### 2.3.4 Pareto-based EAs Approaches

These are the approaches that use the concept of Pareto dominance in order to determine the non-dominated solutions. There exists in the literature many metaheuristics for MOPs that use the Pareto-based concept as GA and PSO.

- **Pareto-based GA Approaches**

The algorithms of this approach may be categorized into non-elitist and elitist algorithms.

- **Non-elitist Algorithms**

The algorithms of this approach do not use an elite-preservation operator which retains better solutions through generations. Therefore, promising solutions may not survive during the algorithm run [31]. The following are the most common non-elitist algorithms:

\* **Multi Objective Genetic Algorithm (MOGA)**

This algorithm was introduced by Fonseca and Fleming in 1993 [41]. In MOGA, the individuals are sorted according to their ranks. The rank of each individual is related to the number of solutions by which it is dominated. A fitness value is assigned for each individual based on its rank with non-dominated solutions having highest fitness value. Individuals with the same rank will share the same fitness value. To maintain diversity, a niching mechanism has been introduced to the algorithm [31][100]. Although the fitness assignment procedure is simple, the algorithm requires a large population size in order to obtain good results. Also, an adjustment of the sharing factor value or the sharing radius ( $\sigma$ ) is required as it affects the performance of the algorithm [31]. The sharing factor or the sharing radius ( $\sigma$ ) defines the threshold of dissimilarity in the niche such that individuals with this radius ( $\sigma$ ) will be considered similar to each other and share fitness.

\* **Niched-Pareto Genetic Algorithm (NPGA)**

This algorithm was proposed by Horn et al. [52][53] in 1993,1994. NPGA incorporates a tournament selection based on the concept of Pareto dominance that differs from the selection methods used in VEGA, MOGA and NSGA. With this Pareto domination tournament scheme, two randomly chosen individuals are compared against a subset of the population of size around 10% of the population called (*tdom*) in order to specify whether they are dominated or not. The one that is non-dominated by the subset is then selected. If both are non-dominated or both are dominated, then the tournament result is determined by fitness sharing [31][40]. The selected parents from the Pareto domination tournament scheme will then be used

to create the next population. The main advantage of this algorithm is that no explicit fitness assignment procedure is needed. In addition, the algorithm is efficient in solving many objective optimization problems. The drawback of this approach is that the manner of tuning the two parameters  $tdom$  and  $\sigma$ , has a great influence on the algorithm performance [31].

\* **Non-dominated Sorting Genetic Algorithm (NSGA)**

This algorithm was developed by Srinivas and Deb in 1994 [101]. In NSGA, the population is classified into subpopulations (fronts) based on the ordering of Pareto dominance. The fronts are found in the following way: the second front is found from what remains after the first front is removed, and the third front is found from what remains after the second front has been removed etc. The fitness assignment is performed according to the non-dominated sorting fronts. So that, solutions belonging to the first non-dominated front will be assigned the highest fitness value. In addition, solutions in the same front will be assigned the same fitness value. For this reason, a diversity maintenance scheme is required. In NSGA, The diversity is maintained by a fitness sharing strategy that is applied to each non-dominated front. A roulette wheel selection mechanism is incorporated to select the parents for creating the next generation. The advantage of this algorithm is the simple fitness assignment procedure that is based on non-dominated set. The disadvantage of this approach is the required tuning of  $\sigma$  share parameter which is basically a user defined parameter that has high influence on the sharing function performance [31].

According to [24], a comparative analysis of MOGA, NPGA, NSGA and VEGA in [27] [112] shows that MOGA exhibits the best performance followed by NPGA, NSGA and VEGA. Another comparative study of VE-

GA, NPGA, NSGA and weighted aggregation function was done by Zitzler et al. [120] on multi-objective 0/1 Knapsack problem. The results show that NSGA achieved the best performance followed by VEGA. For NPGA and weighted sum aggregation function, NPGA performed better in the two objective case. Whereas, weighted sum performed better in the three-objective case.

The fundamental disadvantage of this generation (Non-elitist GA algorithms) is its inability to retain promising intermediate solutions during the algorithm run due to the absence of the elite preservation operator. As a result, non elitist algorithms perform worse than elitist algorithms [31].

#### – **Elitist Algorithms**

The elitist algorithms incorporate an elite operator in their procedure. By the elite operator, the algorithm search capability can be improved. Therefore, elitist algorithms could offer better solutions and guarantee better convergence to the Pareto front [31]. The most common elitist algorithms are:

##### \* **Strength Pareto Evolutionary algorithm (SPEA) and SPEA2**

This algorithm was suggested by Zitzler et al. [120] in 1998. The basic idea of this algorithm is to adopt the elitism mechanism by preserving all non-dominated solutions found along the algorithm run in an external archive called external population. At each generation, the current population is combined with the external population. The set of non-dominated solutions result from the current population is added to external population archive in a way that keeps the archive domination free. As such, the external population will always contain the non-dominated solutions found thus far. The result non-dominated solutions of the mixed population are as-



signed fitness values according to the number of solutions they dominate. The fitness assignment procedure is done in two steps. First, the individuals in the external population are assigned fitness values called strength. Then, the individuals in the current population are evaluated based on the strength values of the external solutions. A clustering technique was applied to fix the external population size and to maintain diversity [24][31]. The algorithm is characterized by its simple assignment procedure. Besides, the used clustering algorithm has no additional parameter to fix. However, the drawback of this approach is to find the appropriate size of the external population [31].

An improved version of SPEA called SPEA2 is proposed by Zitzler et al. [118]. This approach differs from the previous one in that it incorporated an enhanced fitness assignment procedure that records for each solution the number of individuals it dominates and it is dominated by, a nearest neighbour density estimation scheme in order to guide the search process more efficiently, and a new archive truncation method to avoid the loss of boundary solutions and maintain good diversity.

#### \* **Non-dominated Sorting Genetic Algorithm II (NSGAII)**

This algorithm was introduced by Deb et al. [32]. In NSGAII, a crowding comparison operator is used to maintain diversity of population along the Pareto optimal front without any additional parameters. The algorithm works on two populations, the first population is created at the beginning of the algorithm and the second population is the offspring. As in NSGA, the algorithm uses a fast non-dominated sorting method for sorting the population into fronts. Each solution is ranked based on the number of

solutions it dominates and the set of solutions by which it is dominated. The crowding distance value is then computed for each solution in each front. This crowding distance operator measures the density of solutions surrounding a given solution in the population. The population needs to be sorted according to each objective function value before using the crowding distance mechanism. Thereafter, the crowding distance estimation procedure selects the solutions according to both, their non-domination rank and their crowding distance values.

The advantage of NSGAII algorithm is that the crowding comparison operator used to maintain diversity between solutions requires no additional parameters. The drawback of this approach is the  $2N$  population size that is required to perform the non-dominated sorting [31].

- **Pareto-based Evolutionary Strategy Elitist Approach**

Pareto Archive Evolutionary Strategy (PAES) has been proposed for this approach.

- **Pareto Archive Evolutionary Strategy (PAES)**

This algorithm was suggested by Knowles and Corne [66]. It is based on a simple  $(1 + 1)$  evolutionary strategy combined with an external archive of all non-dominated solutions found so far by the algorithm. PAES creates a random solution (parent) which is evaluated and added to the archive. The parent solution is then mutated to create a new solution (offspring). The offspring solution is evaluated and compared to the external archive solutions. If the offspring solution dominates any member in the archive, then the offspring solution is added to the archive and the solutions dominated by the offspring are deleted from the archive. For maintaining population diversity along the front, an adaptive grid is applied. Each solution is located in a certain grid or

hypercube according to its objective values [24][31]. Solutions that are located in less crowded hypercubes tend to be selected for the next population. The main advantage of this approach is the use of the adaptive grid that has direct control on the population diversity. The drawback of this approach is the difficulty in finding the appropriate archive size [31].

Grosan et al. [45] proposed a comparative study of the most commonly used algorithms (SPEA, PAES, and NSGAI) based on five test functions introduced by Deb, Zitzler, and Thiele (1999). For test function (T1) and (T2) where the Pareto optimal front is convex and nonconvex respectively, all algorithms show the same performance. For test function (T3), all algorithms exhibit a good approximation of the Pareto front. For test function (T4), NSGAI and SPEA were capable to converge to the true Pareto front. For test function (T5), PAES gave good results. Skolpadungket et al. [100] applied various multi-objective genetic algorithm techniques (VEGA, Fuzzy VEGA, MOGA, SPEA2, and NSGAI) to solve portfolio optimization with some constraints. The results show that NSGAI gives better approximation of the Pareto front than SPEA2. SPEA2 and MOGA performed better in terms of maintaining diverse solutions.

- **Pareto-based PSO Approaches**

There are different Pareto-based MOPSO approaches proposed in literature for selecting the suitable guide. The guide is the particle that is used to direct another particle in its journey towards better areas of the search space. The guide is also called global best position or leader. Most of these methods are based on some density measures calculated in the objective space in order to choose the leader [6]. In this section we present the most important selection methods with a brief description of their properties. A comprehensive review of PSO algorithms for solving MOPs can

be found in [92][82].

### **Different Variations of Leader Selection Schemes**

Coello et al. [20] proposed one of the earliest Pareto-based multi-objective PSO approaches (MOPSO). An external archive is used to store the non-dominated solutions found during the search process. This external archive is of fixed size and is maintained by giving priority to particles located in less crowded regions of the objective space. The search space is divided into hypercubes inspired by the adaptive grid of PAES approach [66]. A roulette wheel selection scheme is used to select a hypercube and then a global best (leader) is selected randomly from the chosen hypercube. Mutation has been incorporated into the algorithm to maintain diversity. The algorithm is compared against PAES and NSGAII using two test functions. The experimental results show that MOPSO outperforms the other competing algorithms on one test function and gives similar results on the second test function. This approach deals only with bi-objective optimization problems because the number of non-dominated solutions increases greatly with the number of objectives. Subsequently, the update process of the adaptive grid, which stores the non-dominated solutions, becomes more difficult and time consuming as it needs to be maintained at each iteration.

Fieldsend et al. [38] proposed an approach similar to MOPSO [20] except for the external archive size. This approach overcomes the disadvantage of the limited archive size of previous proposals by using an unconstrained archive called dominated tree. The global best of each particle is selected based on the closest member of the dominated tree archive to the given particle. The algorithm uses a mutation operator to promote diversity. Like MOPSO [20], Fieldsend algorithm is suitable for

only two objective functions. This is because with many objectives, the dominated tree archive size gets large due to the increase in the number of non-dominated solutions. As a result, the update operation of the dominated tree archive becomes too difficult and time consuming.

A new density measure scheme called sigma method is proposed by Mostaghim and Teich [75] to select the global best leader for each particle. This approach is called (Sigma-MOPSO). Each particle in the swarm is assigned a sigma value as well as all the archive members. To select a *gbest* for a particle, the external archive member with the closest sigma value of the given particle is selected as its *gbest* guide. The algorithm uses a mutation operator to maintain diversity. The algorithm requires a large population size to obtain well distributed solutions in order to find the best guide for each particle in the swarm. Besides, The sigma value may cause a premature convergence in some cases [92]. When compared with the dominated tree of Fieldsend et al. [38] on four test functions, the sigma method outperformed the dominated tree method. This approach has been successfully tested on a molecular force field problem giving promising results. In a further proposal, Mostagim and Teich [74] studied the effect of the archiving method  $\varepsilon$ -dominance on multi-objective PSO (MOPSO). The authors use the  $\varepsilon$ -dominance strategy in order to bound the archive size. This proposed archiving method is compared to the existing clustering archiving approach [79] giving promising results in terms of computation time, convergence and diversity. The clustering method is an archiving technique that is used to maintain the archive size of the non-dominated solutions. It will be described later in section 6.1.1.

Another new mechanism for selecting the global best leader called stripes is proposed by Villalobos et al. [113]. The stripes are applied on the objective space in order to select a leader and to promote diversity as well. The algorithm is compared against NSGAII and another state-of-the-art approach ( $\epsilon$ MOEA) that is based on  $\epsilon$ -dominance concept. The experimental results show that the proposed algorithm is a good performer. A different approach is proposed by Huo et al. [57] called (Smart-MOPSO). The basic idea of this approach is to evaluate each particle with respect to each objective function independently. The selected global best of the swarm is the mean of the best particles per objective function. The algorithm is not fully evaluated with respect to other methods.

A new hybridized approach that introduces NSGAII non-domination sorting mechanism [32] into PSO algorithm has been proposed by Li [68] and termed as Non-dominated Sorting PSO (NSPSO). As NSGAII works with  $2N$  population, the algorithm combines the self best positions of all particles in the swarm with all the new positions to form the  $2N$  population. The non-dominated solutions within this population are recorded and sorted according to a niche count or crowding distance values. Then for each particle in the swarm, the global best leader is selected randomly among the top part of the sorted list of the non-dominated solutions. The size of this top part is user-specified (5%, 10%, ...). A mutation operator is also used in NSPSO to promote diversity. The experimental results show that NSPSO is highly competitive when compared with NSGAII in terms of convergence and distribution of solutions. Another approach by Li [69] called MaximinPSO which employs the maximin fitness function proposed by Balling [8] in order to obtain the non-dominated solutions. As the algorithm adopts the maximin function, it does not require any additional niching or clustering schemes for maintaining popula-

tion diversity. The non-dominated solutions are stored in an external archive and sorted according to their maximin values. For each particle, the leader is selected randomly among the top part of the sorted external archive. The number of candidates in this part is user-specified. MaximinPSO is compared with NSGAII on the ZDT test function series. The ZDT is a popular test suite created by Zitzler et al. [117]. Each of the ZDT test functions contains a particular characteristic that could cause difficulty in converging to the true Pareto front. The results show that MaximinPSO outperforms NSGAII in terms of convergence, time complexity, and diversity. However, MaximinPSO is tested only on unconstrained problems [69].

In a further work, Bartz et al. [13] studied the influence of elitist archiving on the performance of PSO in a multi-objective context. Each particle is assigned a selection and deletion fitness values calculated by using two functions  $F_{sel}$  and  $F_{del}$  respectively. An archive member is selected as a leader based on  $F_{sel}$  in a roulette wheel selection scheme. On the other hand, an archive member is selected for deletion based on  $F_{del}$  when the archive is full. The method was analysed thoroughly to demonstrate the good performance of the proposed approach. Furthermore, Alvarez et al. [6] proposed three different leader selection techniques that are based on Pareto dominance concepts, namely Rounds, Random, and Prob. Each technique maintains a specific feature in the algorithm. For instance, Rounds maintains diversity, Random maintains convergence, and Prob is a combination of the previous two techniques. The algorithm also handles constraints concluding that regions near constraint boundaries have to be explored properly in order to ensure convergence to the Pareto front.

In recent work, Wickramasinghe Li [114] proposed a new hybrid particle swarm optimization algorithm (MDEPSO) which uses a Differential Evolution (DE) operator to create a diverse range of leaders. By this feature, the algorithm was able to overcome the premature convergence problem when solving problems with many local optimal fronts. The algorithm has been shown to be very competitive when compared to NSGAII and other existing multi-objective PSO algorithms. Thereafter, Jiang et al. [61] presented a novel method to maintain the external archive and to select the global guide for each particle in each iteration. The method divides the non-dominated solutions in the archive into two types based on the dominance relationship among archive members. The first type is the non-dominated solutions without any domination to enhance convergence. The second type is the non-dominated solutions with domination to improve diversity. The algorithm shows competitive results when compared to other multi-objective evolutionary algorithms on three test problems [114].

Moreover, Pang et al. [80] proposed a novel MOPSO algorithm that adopts a new leader selecting strategy which is based on entropy information. The entropy value is evaluated for each particle, then the particle from the Pareto optimal set with a higher entropy value will be selected as the leader for the current particle using the roulette wheel selection scheme. With this new leader selection strategy, the algorithm will maintain good convergence to the Pareto front and maintain good diversity of the obtained Pareto optimal solutions. In addition, an adaptive chaotic mutation operator is adopted in order to avoid premature convergence. The proposed algorithm is compared with two existing PSO multi-objective algorithms on the six benchmark functions, namely ZDT1-4, DTLZ1 and DTLZ2. The results reveal that the proposed algorithm outperformed the other two algorithms.



### **PSO Combined with Crowding Distance Scheme**

The leader selection mechanism adopted in the algorithm proposed by Ray et al. [90] is based on the crowding radius-based mechanism combined with a roulette wheel selection scheme to maintain diversity. The set of non-dominated solutions is determined by using a multi level ranking strategy that ranks solutions by the Pareto ranking scheme. The Pareto ranking scheme is also used as the handling constraint procedures in this method. The algorithm is validated on two test functions and on an engineering design optimization problem. The algorithm exhibits competitive results when compared to NSGAI.

Similar to Ray et al. [90], Raquel et al. [89] proposed an approach called (MOPSO-CD) that adopts the crowding distance scheme for selecting the suitable leader for each particle, as well as for deleting leaders from the non-dominated external archive when it is full. Each non-dominated solution is assigned a crowding distance value. For leader selection, the non-dominated external archive is sorted in terms of the crowding distance value in descending order and the particle's leader is selected randomly from top of the archive. On the other hand, when the external archive size exceeds the threshold, it is also sorted in descending order according to leaders' crowding distance values and the archive member to be deleted is selected randomly from the bottom of the archive. Mutation is incorporated to the algorithm for diversity promotion. The constraint handling technique from NSGAI [32] is also employed to solve constrained problems. The algorithm has been shown to be highly competitive when compared to MOPSO [89].

## Two External Archives Approaches

In a further proposal, Branke and Mostaghim [17] studied the impact of archiving the personal best positions in MOPSO. In this approach, a personal archive is used for each particle to keep all the personal best solutions found during its journey. Different techniques from literature are adopted in order to select the *sbest* particle from the personal archive such as Random selection and diversity preservation. Such techniques are compared on some benchmark test problems to demonstrate the big influence of selecting the suitable self best on the algorithm performance.

Following the same principle, Sierra and Coello [99] used two external archives in their approach. One of the archives is used to store the best positions of the current generation of the algorithm. While the other archive is used to store the final non-dominated solutions. The algorithm adopts the crowding distance estimator for leader selection and leader deletion with respect to their crowding values. In addition, mutation and  $\varepsilon$ -dominance are incorporated for diversity promotion and non-dominated archive bounding respectively. The proposed algorithm has shown to outperform three other MOPSO algorithms and to be highly competitive to NS-GAII and SPEA2.

Like [17] and [99], Ho et al. [51] proposed a PSO-based vector algorithm for multi-objective design problems. In this approach, a novel formula for updating the velocity and position of each particle is introduced in order to improve the global search capabilities of PSO. In addition, a craziness operator is proposed to maintain population diversity. This approach uses two external archives, one is global for the whole swarm and the other is local for each particle to preserve the most recent non-dominated solutions it has encountered. During the particle's update procedures, the

particle selects its personal best position from the local archive and the global best position from the global archive using a roulette wheel selection scheme. The algorithm is validated on two test functions with no comparison to other algorithms [51].

Another similar approach is presented by Abido [1]. This approach uses two external archives local and global. The basic idea of this approach is to store not only the non-dominated solutions obtained during the search process, in order to select the global best solution, but also to store the self best solutions found by each particle in order to select its local best solution. The proposed method has been tested on four test problems and compared with SPEA. The results show the superiority of the proposed method over SPEA.

### **Sub-Population Approaches**

Unlike the previous approaches that adopt two external archives [17][99][51][1], Pulido and Coello [86] developed a new MOPSO called another multi-objective particle swarm optimization (AMOPSO) which is based on Pareto ranking and clustering algorithm. The main idea of this approach is to divide the decision variable space into several subswarms using the clustering mechanism in order to explore more regions of the search space and to maintain diversity. Each subswarm has its own group of leaders from which one is selected randomly to guide the particle in its flight. At a certain predefined point, the subswarms exchange information through leaders migration. With this feature, the algorithm does not need to use an external archive. The algorithm has been shown to be very competitive with state of the art multi-objective evolutionary algorithms. In a further work, Pulido et al. [109] developed an enhanced version of AMOPSO [86] called Efficient MOPSO (EMOPSO). In order to maintain diversity, the authors proposed a mechanism called Hyper-plane

distribution in addition to  $\varepsilon$ -dominance and adaptive grid archiving methods. A turbulence operator is also incorporated to avoid premature convergence. Constraints are handled using a strategy already proposed by Coello and Pulido in [85]. The algorithm was able to achieve promising results with a very small number of fitness function evaluations [109].

Similarly to AMOPSO [86], Janson and Merkle [60] proposed a new hybrid multi-objective PSO called (ClusterMPSO). The hybridization involves the incorporation of PSO algorithm into the clustering K-means algorithm in order to split the population into several subswarms. Each subswarm has its own set of leaders. The final set of leaders is the union of all subswarms' set of leaders. The algorithm is tested on artificial optimization functions as well as on a real world biochemistry problem with promising results.

Another approach following the subswarm concept is proposed by Mostaghim and Teich [76]. In this approach, an algorithm for covering the Pareto front called (covering MOPSO) is proposed. The algorithm has two phases. The first phase uses a restricted archive MOPSO method to find the non-dominated solutions that are very close to the Pareto front. In the second phase, the population is divided into subswarms that are initialized around the non-dominated solutions obtained from phase one using the covering MOPSO with unbounded archive. As such, the search process will be limited to the neighbourhood around each non-dominated solution. The algorithm outperformed an existing MOEA covering method called Hybrid MOEA when both applied to a real world antenna design problem.

## 2.4 Inadequacies of Previous Work

From the above literature, we can see that there exist several methods for solving MOPs. Now we are going through the methods in Figure 2.6 and highlight the advantages and disadvantages of the various approaches.

The classical methods (**weighted sum aggregation and lexicographic ordering**) convert the multi-objective problem into a single objective problem using scalar optimization techniques. For instance, the weighted sum aggregation approach converts the MOP into a SOP by combining all the objectives into a higher scalar function based on prior information of the problem. Although this method is easy to implement, selecting the weights of the objectives is not an easy task, there is no straightforward way to do it because it requires a profound domain knowledge, which is often not available for most real-world problems. The determination of these weights becomes even more difficult as the number of objectives increases. Furthermore, it produces only a single compromise solution per run based on the selected weights. As such, the algorithm has to be run repeatedly in order to obtain the set of non-dominated solutions. In addition, this approach does not work well with non-convex Pareto fronts [81], regardless of the weights used.

The lexicographic ordering approach treats MOP as SOP by optimizing a single objective at each run of the algorithm according to its order of importance. This approach is easy to implement. However, it requires prior information from the user to rank the objectives in order of importance. This pre-defined ordering of objectives is very important as it affects the performance of the algorithm. A good ordering is even more difficult to establish with many objectives. Also this method generates one single solution at each run like the weighted sum aggregation methods. Generally, these classical methods are

well suited for cases where prior knowledge about the objectives is known.

Unlike the classical methods, the **Vector Evaluated Genetic Algorithms (VEGA)** approach is able to generate multiple non-dominated solutions in parallel. Besides, it is easy enough to implement. However, this approach suffers from a bias towards some solutions that excel in only one of the objectives. This fact prevents the algorithm from generating compromise solutions with regard to all objectives. Another weakness of this approach is its inability to solve non-convex Pareto fronts [40].

The three approaches mentioned above (weighted sum aggregation, lexicographic ordering, and VEGA) do not make direct incorporation of the actual definition of Pareto optimality. In other words, the concept of trade off between objectives is missing. That is why they are called Non-Pareto based approaches. These approaches are susceptible to the shape of the Pareto front, for example it cannot find non-convex fronts [81][40]. This weakness of the search efficiency make their direct use inappropriate for handling many objectives, since in the case of many objectives there is a high chance that non-convex Pareto fronts arise.

**Pareto based evolutionary algorithm approaches** compare solutions according to the Pareto dominance relation, i.e., the concept of Pareto dominance is applied, in order to find the set of high fitness non-dominated solutions in the population. **Pareto-based GA approaches** require a Pareto ranking procedure to direct the search towards the Pareto front. They also require a diversity preserving mechanism (niching) to maintain diversity in the population and prevent the GA from converging to a single solution. Although they are relatively easy to implement, their main drawback is that their performance depends highly on the selection of the sharing factor.

On the other hand, we can see that the bulk of work of **Pareto based PSO approaches** has concentrated on three main directions:

- The first direction consists of developing an effective leader selection strategy for global best particle and self best particle. Most of these selection methods are based on a niching technique, random selection, and some density measure methods such as the crowding distance scheme [92][79].
- The second direction deals with developing an efficient archiving mechanism to store the non-dominated solutions over iterations
- The third direction is the need to use a diversity preserving mechanism in order to obtain a diverse set of non-dominated solutions.

We identified a number of limitations of multi-objective PSO:

- Although a great deal of effort has been devoted to solve MOPs with PSO, the problem is still open in a sense that till now the approximation to the Pareto front still requires improvement.
- PSO uses several tunable parameters ( $W$ ,  $c_1$ , and  $c_2$ ). Finding the best setting of those parameters is a time consuming process.
- Most of MOPSO algorithms are combined with a mutation (turbulence) operator that has been shown to promote diversity. However, choosing a good mutation operator is a non easy task. For instance [92], a decision should be made on the component of the particle that has to be mutated and the probability of mutation.

To recapitulate, the Pareto based evolutionary algorithms (metaheuristic techniques) overcome the limitations of the Non-Pareto based approaches when generating the Pareto

front. First, they can obtain the set of non-dominated solutions in a single run as they can allow simultaneous exploration of different points of the Pareto front. Second, they can perform the optimization process without any prior knowledge about the problem. Third, they are not susceptible to the shape or continuity of the Pareto front. However, their main weakness is that their performance degrades when the number of objectives to be optimized increases. The Pareto based GA approaches need computationally efficient methods for performing the Pareto ranking procedure as it has to be repeated over and over during the evolutionary process. Besides, they are very sensitive to the value of the sharing factor. When dealing with many objectives, the performance of the Pareto based PSO approaches relies heavily on the leader selection strategy being used. For instance, the MOPSO approach proposed by Coello [20] used the adaptive grid scheme to select the suitable guide for each particle. This technique does not scale well with an increasing number of objectives because the update of the adaptive grid becomes more difficult and time consuming.

Many objectives lead to further difficulties with respect to decision making, visualization, search efficiency and computational cost [94]. Decision making and visualization remain easy to perform with aggregation based methods and lexicographic methods unlike the other methods that provide a set of non-dominated solutions per run. The challenge of search efficiency and computational cost becomes even more important for all of them to solve many objective problems. Nowadays, the scalability of methods and the design of new methods to handle many objective problems is becoming a very hot topic.

## 2.5 Summary

In this chapter, we have introduced the basic definitions, notations, and concepts associated with multi-objective optimization, particle swarm optimization and quantum



behaved particle swarm optimization. Then we presented an overview of the different multi-objective evolutionary algorithms that have been developed and successfully applied to solve multi-objective optimization problems. We emphasized the PSO-Pareto based methods because these are most relevant for the remainder of the thesis, and they represent the current state of art. The chapter ends with a summary of the important strengths and weaknesses of the different MOP approaches.

In the following chapter, a new addition to the canyon of swarm based multi-objective optimization methods will be devised. Following the same spirit of MOPSO, we propose an extension of QPSO to solve continuous multi-objective optimization problems that aims to achieve better convergence and diversity simultaneously. Our new approach is termed in the following as Multi Objective Quantum-behaved Particle Swarm Optimization (MOQPSO). It is shown in Figure 2.6 with the dashed box.

### **A New Framework for MOP: Multi-Objective Quantum-behaved Particle Swarm Optimization (MOQPSO) for Unconstrained Problems**

---

In this chapter,<sup>1</sup> we propose a new approach for multi-objective optimization based on QPSO. In particular, we show how we extended QPSO, and developed it to handle unconstrained multi-objective optimization problems. This extension includes a definition of a leader selection strategy, a policy to maintain the Pareto set during the search process and an overall dynamics that helps evolving an initial Pareto set towards the optimal one. Specifically, we address the way the global best solutions are recorded within an archive and used to compute the local attractor point of each particle.

---

<sup>1</sup>A shorter version of the work in this chapter has been published in the following: Heyam Al-Baity, Souham Meshoul, and Ata Kaban. On Extending Quantum Behaved Particle Swarm Optimization to Multi-objective Context. In Proceedings of the IEEE World Congress on Computational Intelligence (IEEE CEC 2012), pp. 1-8, 2012.

### 3.1 Main Features of the Proposed MOQPSO

The typical dynamics of a multi-objective Pareto based swarm algorithm include two main phases. In the first phase, the initial swarm is generated, the algorithm parameters are set and an initial set of non-dominated solutions is derived. The second phase is generally an iterative procedure during which the positions of particles are recomputed according to equations (2.3) and (2.4) as explained in section 2.2.2. Self best performance of each particle is updated along with the current Pareto set of non-dominated solutions. Therefore, extending QPSO to multi-objective optimization should be done in a way to find a Pareto front as close as possible to the optimal one while ensuring a uniform distribution of the non-dominated solutions within it. We refer to the proposed approach as MOQPSO (Multi-Objective Quantum-behaved Particle Swarm Optimization). Generally, in single objective QPSO, only one self best particle and one global best particle have to be considered when updating particles' positions. In a multi-objective context, a set of non-dominated solutions should be handled. Recall the main equation (2.3) in section 2.2.2, that governs the move of particles in the search space:

$$x_{ij}^{t+1} = p_{ij}^t \pm \beta \cdot |mbest_j^t - x_{ij}^t| \cdot \ln(1/u_{ij}^t)$$

We can identify from this equation, in a direct way, the two main channels through which the extension of QPSO to MOP should be studied, namely: the local attractor ( $p_{ij}^t$ ) and the mean best ( $mbest$ ). In an indirect way, the impact of the contraction expansion parameter  $\beta$  has to be investigated as well. Therefore, the computation of each particle position requires three main components namely:

- The local attractor point of a particle. As given in equation (2.4) in section 2.2.2:

$$p_{ij}^t = \varphi_{ij}^t \cdot sbest_{ij}^t + (1 - \varphi_{ij}^t) \cdot gbest_j^t \quad \text{with } \varphi_{ij}^t = rand(0, 1)$$

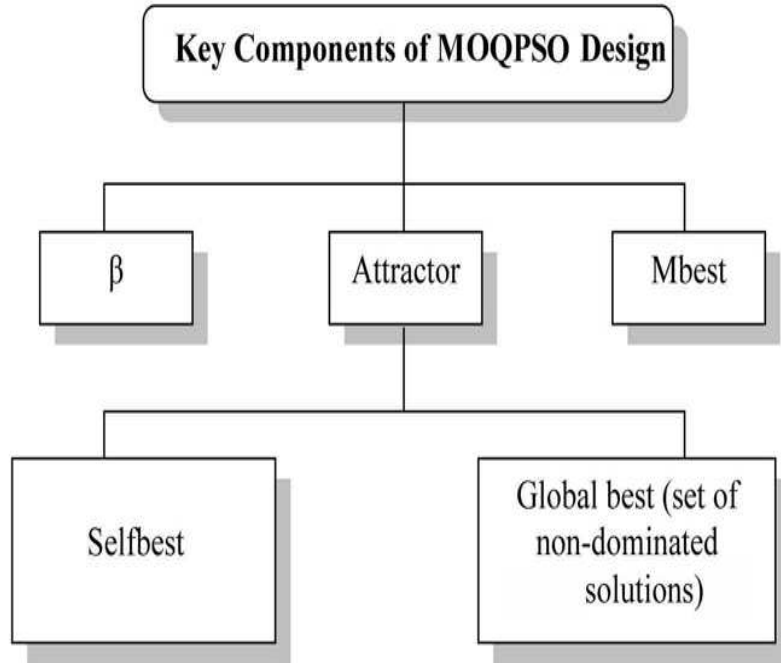
It is calculated in a way that balances the influence of the self best performance of the particle and the global best performance of the entire swarm.

- The mean best position (also called mainstream thought by Sun et al. [105]), which is obtained by averaging all the self best positions as shown in equation (2.5) in section 2.2.2:

$$mbest^t = \frac{1}{N} \sum_{i=1}^N sbest_i = (\frac{1}{N} \sum_{i=1}^N sbest_{i1}, \frac{1}{N} \sum_{i=1}^N sbest_{i2}, \dots, \frac{1}{N} \sum_{i=1}^N sbest_{iD})$$

- The contraction expansion parameter  $\beta$

Figure 3.1 below illustrates the key components that influence the design of multiobjective QPSO (MQPSO).



Figur 3.1: The key components in MOQPSO design

Another way to highlight the main features of the proposed MOQPSO is through the differences between single objective QPSO and the proposed multi-objective QPSO

algorithm in terms of the number of self best and global best particles, the update strategy, the selection strategy, and the *mbest*. These differences are summarized in Table 3.1.

Therefore, designing MOQPSO leads naturally to the following important research questions:

- How should the local attractor be updated for a given particle?
- How should the mean best performance be derived?
- What is the impact of the contraction-expansion parameter ( $\beta$ ) on the convergence of the algorithm?
- Which leader to select as the global best position for a particular particle to properly guide the particle's navigation in a D-dimensional search space?
- Which archiving strategy to use to maintain the set of non-dominated solutions during the search process?

## 3.2 Description of the Proposed MOQPSO

Over the next sections, we will describe the proposed framework by bringing answers to the above questions. Our attempts in addressing these issues, while taking into account the proposals available in the literature, have resulted in defining new strategies for leader selection and archiving of non-dominated solutions.

### 3.2.1 Computing Local Attractor Points in MOP

The convergence of an individual particle relies heavily on the local attractor point [103]. It helps achieving a certain balance between the self best performance of the particle and the global best performance as it is a function of both of them. With several objectives, the global best positions and the self best positions should be computed in a different manner than in the single objective case using the Pareto dominance relation.

Tabel 3.1: Comparison between a single-objective and multi-objective QPSO optimizations

Swarm Particles	Single objective context				Multi-objective context			
	Number	Update strategy	Select strategy	Mbest	Number	Update strategy	Select strategy	Mbest
<b>Casual particles</b>	many	At each iteration according to equation (2.3)	None	N/A	Many	At each iteration according to equation (2.3)	None	N/A
<b>Self best particle</b>	One	-For each particle. -At each iteration according to a simple comparison	Straightforward	Average	One	-For each particle. -At each iteration according to dominance relation	Straightforward	Average
<b>Global best particle</b>	One	At each iteration according to a simple comparison	Straightforward	N/A	Many (GBA)	At each iteration according to the archive update strategy	-For each particle, at each iteration according to the leader selection strategy	N/A

- For global best positions, an archive is used to keep the non-dominated solutions found during the search process. We call such an archive the Global Best Archive (GBA). The global best position of each particle is then selected from the GBA. The difficulty arises as all the GBA members are equally important in the sense of Pareto optimality. However, selecting one of them should be properly performed in order to improve both convergence and diversity of the obtained set of non-dominated solutions. The convergence property refers to the ability to achieve fronts that are as close as possible to true Pareto fronts while the diversity property refers to the ability to achieve a good spread of non-dominated solutions along the obtained fronts. The diversity of the set of non-dominated solutions is important in the MOP domain as it can provide better choices to the decision makers. For this purpose, we propose a new selection strategy that is inspired from the sigma method proposed in [75] and the crowding distance information proposed in [32]. A detailed description of this selection strategy is given in the following section.
- For self best position, the strategy we followed in our work is to keep only one solution as a self best point for each particle. Once a new position is computed for a particle, three cases may appear:
  1. The new position dominates the self best position in which case this latter should be updated.
  2. The new position is dominated by the self best position in which case this latter remains the same.
  3. None of the two dominates the other; in this case one of them is randomly selected and kept as the self best position for the particle.
- Finally, once the *sbest* and *gbest* positions are determined for a particular particle, the local attractor can be computed as given in equation (2.4) in section 2.2.2.

### 3.2.2 Calculating the Mean Best Position

As only one solution is kept as the self best position for each particle, the mean best position is computed as in single objective optimization context which is shown in equation (2.5) in section 2.2.2 :

$$mbest^t = \frac{1}{N} \sum_{i=1}^N sbest_i = (\frac{1}{N} \sum_{i=1}^N sbest_{i1}, \frac{1}{N} \sum_{i=1}^N sbest_{i2}, \dots, \frac{1}{N} \sum_{i=1}^N sbest_{iD})$$

### 3.2.3 Setting of the Contraction Expansion Parameter $\beta$

We use a time varying parameter  $\beta$ . It has been shown [37][104] that in the case of single objective optimization, good results have been obtained when decreasing  $\beta$  linearly from an initial value  $\beta_{max}$  to a final value  $\beta_{min}$  during the search process. At each iteration  $t$ , the new value of  $\beta$  is calculated as a function of the current value of  $\beta$  at iteration  $t$ ,  $\beta_{max}$ ,  $\beta_{min}$ , and the maximum number of iterations  $T$ :

$$\beta^{t+1} = \beta^t - \frac{(\beta_{max} - \beta_{min})}{T}$$

Before an in-depth investigation of our proposed selection and archiving strategies, let us first describe the outline of the proposed MOQPSO.

### 3.2.4 Outline of the General Framework of MOQPSO for Unconstrained Problems

A specification of an unconstrained MOP includes the definition of the problem dimension and the fitness functions along with the lower and upper bounds for each decision variable. Let us denote the set of non-dominated solutions or the archive of global best solutions in Pareto sense encountered at iteration  $t$  by  $GBA^t$ .  $S^t$  refers to the swarm of particles at iteration  $t$ . The proposed MOQPSO is described in Algorithm 3.



---

**Algorithm 3** Pseudocode of MOQPSO

---

```
1: Input: MOP specification
2: N= population size
3: D= problem dimension
4:  $S^0$ = Initialize-Positions()
5:  $sbest_i$ = initialize self best position of particle  $P_i$  for  $i=1..N$ 
6: T= maximum number of iterations
7:  $\vec{F}_i$ = evaluate particle  $P_i$  for  $i=1..N$ 
8:  $GBA^0$ = initial set of non-dominated solutions
9:  $t = 1$ 
10:  $\beta^t = \beta_{max}$ 
11: repeat
12:   Compute mean best position using eq. (2.5)
13:   for (each particle  $P_i$  ) do
14:      $gbest = Select - leader(GBA^t, P_i)$ 
15:     for (each dimension j ) do
16:        $p_{ij}^t = \text{Compute local attractor using eq. (2.4)}$ 
17:        $u_{ij} = rand(0, 1)$ 
18:       if  $rand(0, 1) > 0.5$  then
19:          $x_{ij}^{t+1} = p_{ij}^t + \beta^t \cdot |mbest_j^t - x_{ij}^t| \cdot \ln(1/u_{ij}^t)$  for  $j=1..D$ 
20:       else
21:          $x_{ij}^{t+1} = p_{ij}^t - \beta^t \cdot |mbest_j^t - x_{ij}^t| \cdot \ln(1/u_{ij}^t)$  for  $j=1..D$ 
22:       end if
23:     end for
24:     Evaluate particle  $P_i$ 
25:     Update self best position
26:   end for
27:    $GBA^{t+1} = Update - Archive(GBA^t, S^t)$ 
28:    $\beta^{t+1} = \beta^t - \frac{(\beta_{max} - \beta_{min})}{T}$  {Decrease  $\beta$  linearly}
29:    $t = t + 1$ 
30: until ( $t \succ T$ )
31: Output :  $GBA$ 
```

---

At the beginning of the algorithm, the  $N$  particles' positions are initialized with uniformly distributed pseudo-random numbers within the allowable ranges  $[min_j, max_j]$  at each dimension as outlined by the procedure 'Initialize-Positions' below where  $rand()$  refers to a function that returns pseudo-random scalar drawn from the standard uniform distribution in the open interval  $(0,1)$ .  $[min_j, max_j]$  is the range of the  $j^{th}$  decision variable's values. The initial positions of the particles are then assigned to the corresponding self best positions. MOQPSO uses an external unbounded archive called GBA to keep the non-dominated solutions obtained during the search process. At each iteration, the mean best position is computed over the self best positions. Then for each particle  $P_i$ , a leader is selected as the *gbest* position to compute its local attractor and a new position is derived for this particle. The self best position is updated accordingly based on the dominance relation. Note that in our work, we keep only one self best position per particle. Once all particles have been processed, the GBA is then updated with the new set of non-dominated solutions. The behaviour of the algorithm is controlled by decreasing the contraction expansion parameter linearly. High values of this parameter favour exploration capabilities of the search space while small ones foster its intensification capabilities. The termination criterion used in MOQPSO is the maximum number of iterations.

---

**Algorithm 4** Initialize-Positions( )

---

```

1: Input: Population size, Problem dimension, Domain  $[min, max]$  in each dimension
2: for (each particle  $P_i$ ) do
3:   for (each dimension  $j$ ) do
4:      $x_{ij} = min_j + (max_j - min_j) * rand;$ 
5:   end for
6: end for
7: Output : Particles positions  $X = [x_{ij}]$  for  $i=1..N$  and  $j=1..D$ 

```

---

### 3.3 The Proposed Leader Selection Strategy

The main goals of a multi-objective optimization process are to minimize the distance between the obtained Pareto optimal solutions and the true Pareto optimal front and to maximize the spread of the generated Pareto optimal solutions and ensure that they are uniformly distributed along the true Pareto front [31]. In order to achieve the above goals, a suitable guiding of the search process that fosters exploitation and exploration using an effective selection strategy of the leader particle should be adopted. The leader is the particle that is used to guide another particle in its trajectory towards better areas of the search space. The leader is also called global best position or guide. Therefore, one of the key issues when dealing with MOP is how to select a leader or a guide among a set of non-dominated solutions, which are all equally important, in order to guide the swarm in its movement. The concept of “global best solution” in single objective optimization context is substituted by the concept of “set of the non-dominated solutions” in multi-objective optimization. Therefore, selecting the global best guide is a non-trivial task as it has a significant impact on the algorithm performance in terms of convergence and diversity because the global best individual attracts all other particles of the swarm to its direction [31].

In our approach, we propose a new hybrid leader selection method that is based on the combination of both the sigma method [75] and the crowding distance method [32]. The rationale is to help convergence of each particle using sigma values while favouring less crowded regions in the objective space to attain a uniformly spread-out Pareto front using crowding distance values. The proposed method is a two level selection strategy that operates as follows. Given a particle for which we need to select a global leader in order to compute its local attractor point, the procedure first determines the members in the GBA

that are close to the particle in terms of sigma values. Then, the less crowded solution among these neighbours is chosen as the global best solution for the given particle. This principle is illustrated in Figure 3.2.

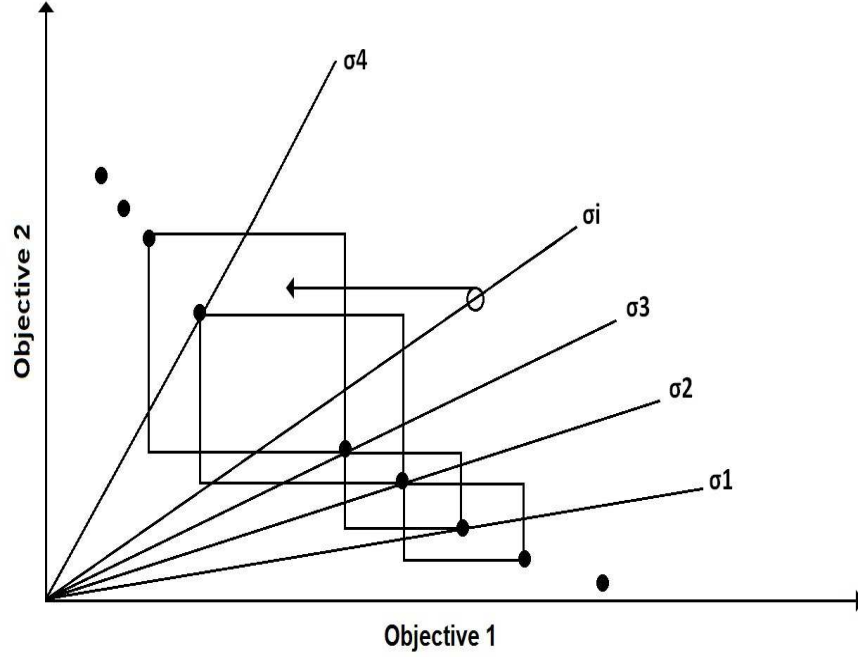


Figure 3.2: Selecting global leader from GBA for  $k=4$ : filled circles are GBA members, empty circle represents particle P

More specifically, it first computes the sigma values of all GBA members as well as of the current particle for which we seek a leader. Then, the  $k$  members from the GBA that are close to the particle in terms of sigma value are selected and sorted based on their crowding distance values. The less crowded solution among the  $k$  ones is then selected as the leader for that particle. More formally, the proposed selection method is described in Algorithm 5.

---

**Algorithm 5** Select-Leader(GBA, P)

---

```
1: Input: Current archive GBA, Current particle P
2: for i=1 to |GBA| do
3:   Sigma(i)= Compute-Sigma-Value(GBA(i));
4: end for
5: Sigma-P = Compute-Sigma-Value(P);
6: Neighbors= Find-k-Nearest-Neighbors(k, Sigma, Sigma-P);
7: Sorted-Neighbors= Sort(Crowding-Distance(GBA), Neighbors, Descending);
8: leader = Sorted-Neighbors(1);
9: Output : leader
```

---

As described in [75], a sigma value is a metric that measures the closeness of particles in the objective space. The sigma value of a particle  $P_i$  defines the line joining the corresponding point in the objective space to the origin. The closeness of two sigma values is in fact an indication that the two corresponding particles lie on two lines that are close to each other. This fact is used to guide the particle by the suitable leader. That is why the k nearest neighbors are selected according to the ascending order of the distance between their sigma value and the particle's sigma value. The sigma value for any particle  $P_i$  whose corresponding point in the objective space is the vector  $\vec{F} = (f_{i1}, f_{i2}, \dots, f_{im})^T$  is given by:

$$\vec{\sigma} = \left[ \begin{array}{c} f_{i1}^2 - f_{i2}^2 \\ \dots\dots \\ f_{i1}^2 - f_{im}^2 \\ f_{i2}^2 - f_{i3}^2 \\ \dots\dots \\ f_{i2}^2 - f_{im}^2 \\ \dots\dots \\ f_{i(m-1)}^2 - f_{im}^2 \end{array} \right] / \sum_{j=1}^m f_{ij}^2$$

The crowding distance is a density estimation method that describes the density of particles that are around a particular particle. It is used to maintain diversity in the Pareto optimal solutions. The crowding value is computed in a similar way as in NSGAII [32]. Solutions in the GBA are first sorted in the objective space according to each objective function in ascending order based on their objective function value. The crowding distance value of each particle for each objective is set as the average distance of its two closest particles, then the final crowding distance value is computed as the sum of the particle crowding distance values with respect to each objective function. The following is the description of the crowding distance scheme [32].

---

**Algorithm 6** Crowding-Distance(GBA)

---

```

1: Input: Current archive GBA
2: for i=1 to |GBA| do
3:   Distance(i) = 0 {initialize distance for each member in GBA}
4: end for
5: for m=1 to M do
6:   GBA = sort (GBA, m) {sort using each objective value}
7:   Distance(1) = Distance(|GBA|) =  $\infty$  {boundary points are always selected}
8:   for i = 2 to (|GBA|-1) do
9:      $Distance(i) = Distance(i) + (GBA(i+1) \cdot objective(m) - GBA(i-1) \cdot objective(m)) / (fmax_m - fmin_m)$ 
       { $fmax_m$  and  $fmin_m$  are the max. and min. values of objective m}
10:   end for
11: end for
12: Output : Distance

```

---

### 3.4 Archiving Mechanism

As described earlier, the algorithm maintains an external archive (GBA) of non-dominated solutions. At each iteration of the search process, the set of non-dominated solutions needs to be updated and maintained dominance-free since new positions have been calcu-

lated. Therefore, the new particles become potential candidates for joining the set of non-dominated solutions in GBA. Each new particle should be compared with each member of the GBA using the Pareto dominance relation. In case the particle is dominated by any GBA member its candidature is discarded. If not, it should be added to the GBA with the removal of all GBA solutions that it dominates. The following is the outline of the proposed update archive procedure.

---

**Algorithm 7** Update-Archive(GBA, S)

---

```

1: Input: Current archive GBA, Current swarm S
2: for (each particle  $P_i$  in S) do
3:   Non-Dominated= TRUE
4:   for (each  $P_j$  in GBA) do
5:     if  $\vec{F}_i$  dominates  $\vec{F}_j$  then
6:       GBA= GBA -  $\{P_j\}$  {  $P_i$  dominates  $P_j$ , therefore  $P_j$  should be removed from GBA}
7:     else
8:       if  $\vec{F}_j$  dominates  $\vec{F}_i$  OR  $P_i = P_j$  then
9:         Non-Dominated= FALSE
10:        Break
11:      end if
12:    end if
13:  end for
14:  if Non-Dominated then
15:    GBA= GBA +  $\{P_i\}$ 
16:  end if
17: end for
18: Output : GBA

```

---

### 3.5 Time Complexity Analysis of MOQPSO for Unconstrained Problems

As described previously in section 3.2.4, the proposed MOQPSO evolves through two stages: an initialization stage followed by an iterative process. In order to analyze the efficiency of the algorithm, we consider the following parameters that impact the size of the problem:

- D: the problem dimension that is the number of decision variables
- M: the number of objective functions
- N: the population size and
- L: the archive size

The main computation during the initialization phase consists of the evaluation of the initial positions and the generation of the initial Pareto front. For each position,  $M$  values of objective functions need to be computed. Therefore the number of functions evaluations is  $O(NM)$ . It is worth to note that the focus is on the number of functions evaluations rather than on the evaluations themselves because the evaluations are performed in the same way whatever the method used. Regarding the generation of the initial front, in the worst case, each solution needs to be compared to all other solutions (the extreme case where the initial solutions are all non-dominated) therefore its time complexity is  $O(N^2)$ . The overall time complexity of the initialization phase is  $O(N^2)$  as  $N$  is largely greater than  $M$  in general.



On the other hand, the main computation in the iterative process encompasses the following tasks:

### 1. Update of Particles' Positions

Since positions are updated for all particles in all dimensions, the time complexity of this task is  $O(ND)$ . Hence, the complexity of the updating step of the algorithm grows in polynomial time with respect to the problem dimension.

For the whole analysis, we assume that the population size parameter  $N$  is bigger than the dimension parameter  $D$ . Therefore, the time complexity of this task becomes  $O(N^2)$ .

### 2. Evaluation of Positions

As in the initialization phase, the number of objective evaluations is  $O(NM)$ . Hence, the complexity of the evaluation step of the algorithm grows in polynomial time with respect to the number of objectives. As can be seen, the dimension parameter  $D$  does not impact the number of objective evaluations but it impacts the evaluation of each objective. The complexity of an objective function evaluation task is related to the function itself rather than to the method used for its optimization.

### 3. Leader Selection

In the leader selection strategy, the main operations consist of three types of sorting:

- Sorting of the archive members according to their crowding distance. This sorting is performed once during each iteration for all particles. Its time complexity is  $O(L \log L)$ .
- Sorting of the archive members according to the objectives during the computation of the crowding distances as described in Algorithm 6, section 3.3.

It requires  $M$  independent sorting operations of the archive according to each objective. This yields to a time complexity of  $O(ML\log L)$  per iteration.

- Sorting of archive members according to their closeness to the current particle in terms of sigma values computed using the values of the  $M$  objectives which yields a time complexity  $O(ML\log L)$  for each particle since the computation of the sigma values is proportional to the number of objectives  $M$ . For all particles,  $N$  independent sorting operations are required. Therefore, the overall time complexity for the leader selection task is  $O(MNL\log L)$ . As the archive size  $L$  is generally proportional to the population size  $N$ , this complexity can be expressed as  $O(MN^2\log N)$ .

- **Update of the Archive or Pareto Front**

Finally in the update archive procedure, each particle in the current swarm needs to be compared to the archive members to decide whether it is non-dominated. At the worst case,  $ML$  comparisons are needed for each particle. Therefore, the overall time complexity of this update archive task is  $O(MNL)$  per iteration. Since the archive size  $L$  is generally proportional to the population size, the time complexity of this task can be expressed as  $O(MN^2)$ .

From the above complexities, it comes out that the estimated time complexity of the proposed framework MOQPSO is  $O(MN^2\log N)$  per iteration, which shows a polynomial complexity with regard to the problem parameters.

### 3.6 Evaluation of the Proposed MOQPSO Algorithm

In the following, we describe the experimental study that has been performed to evaluate the proposed MOQPSO. We first describe the used test problems as well as the performance measures and then the different experiments that have been conducted to study the behaviour of MOQPSO.

### 3.6.1 Test Functions

In order to assess the performance and prove the efficiency of the proposed MOQPSO for solving unconstrained problems, MOQPSO has been applied to various benchmark test problems related to numerical optimization [32]. The difficulties of these test problems range from trivial to non-trivial regarding the ability to find the Pareto optimal solutions and the ability to maintain diversity along the obtained Pareto front. All the used test functions are minimization problems. Their use will allow us to investigate a wide variety of Pareto optimal front characteristics such as convexity, non-convexity, connectivity, disconnectivity, sparsely distributed solutions near the Pareto optimal front and non-uniformity. The true Pareto optimal fronts of these test functions are known. A detailed description of the test problems considered in our experiments is given below.

#### Test Function 1

The first test function is proposed by Schaffer [95]. It has been widely used by multi-objective evolutionary algorithms in order to test how well the non-dominated solutions are distributed along the Pareto front [119]. It is a bi-objective optimization problem with a convex Pareto optimal front. The function is called SCH and is defined as follows:

$$SCH : \begin{cases} \text{Minimize} & f_1(x) = x^2 \\ \text{Minimize} & f_2(x) = (x - 2)^2 \\ \text{where:} & x_i \in [-10^3, 10^3], \quad i = 1. \end{cases}$$

#### Test Function 2

This test function is called FON and is proposed by Fonseca and Fleming [39]. It is a minimization bi-objective optimization problem with non-convex Pareto optimal front. The two objective functions are:

$$FON : \begin{cases} \text{Minimize} & f_1(x) = 1 - \exp\left(-\sum_{i=1}^3 (x_i - \frac{1}{\sqrt{3}})^2\right) \\ \text{Minimize} & f_2(x) = 1 - \exp\left(-\sum_{i=1}^3 (x_i + \frac{1}{\sqrt{3}})^2\right) \\ \text{where:} & x_i \in [-4, 4], \quad i = 1, 2, 3. \end{cases}$$

### Test Function 3

The ZDT1 function is one the ZDT family of functions [117]. It has a convex Pareto optimal front and two objective functions to be minimized:

$$ZDT1 : \begin{cases} \text{Minimize} & f_1(x) = x_1 \\ \text{Minimize} & f_2(x) = g(x) \left[1 - \sqrt{x_1/g(x)}\right] \\ \text{where:} & g(x) = 1 + 9 \left(\sum_{i=2}^n x_i\right)/(n-1) \quad x_i \in [0, 1], \quad n = 30. \end{cases}$$

### Test Function 4

The fourth test function is ZDT2 [117]. This function has a non-convex Pareto optimal front. The two objective functions are:

$$ZDT2 : \begin{cases} \text{Minimize} & f_1(x) = x_1 \\ \text{Minimize} & f_2(x) = g(x) \left[1 - (x_1/g(x))^2\right] \\ \text{where:} & g(x) = 1 + 9 \left(\sum_{i=2}^n x_i\right)/(n-1) \quad x_i \in [0, 1], \quad n = 30. \end{cases}$$

### Test Function 5

The ZDT3 function [117] has a Pareto optimal front that consists of several disconnected convex parts. ZDT3 function is defined as follows:

$$ZDT3 : \begin{cases} \text{Minimize} & f_1(x) = x_1 \\ \text{Minimize} & f_2(x) = g(x) \left[ 1 - \sqrt{x_1/g(x)} - \frac{x_1}{g(x)} \sin(10\Pi x_1) \right] \\ \text{where:} & g(x) = 1 + 9 \left( \sum_{i=2}^n x_i \right) / (n-1) \quad x_i \in [0, 1], \quad n = 30. \end{cases}$$

## Test Function 6

The ZDT6 function [117] has a non-convex Pareto optimal front and the Pareto optimal set is non-uniformly distributed along the Pareto front. In addition, the number of solutions decreases near the Pareto optimal front and increases away from it. Therefore, it is used to evaluate the ability of the algorithm in finding Pareto optimal solutions with good distribution. The objective functions are defined as:

$$ZDT6 : \begin{cases} \text{Minimize} & f_1(x) = 1 - \exp(-4x_1) \sin^6(6\Pi x_1) \\ \text{Minimize} & f_2(x) = g(x) \left[ 1 - \left( f_1(x)/g(x) \right)^2 \right] \\ \text{where:} & g(x) = 1 + 9 \left[ \left( \sum_{i=2}^n x_i \right) / (n-1) \right]^{0.25} \quad x_i \in [0, 1], \quad n = 10. \end{cases}$$

### 3.6.2 Performance Measures

For any multi-objective problem, the two main goals of optimization are to obtain a Pareto optimal set which closely approximates the true Pareto optimal front and to spread out the non-dominated solutions uniformly throughout the Pareto front [31]. In order to assess the ability of the proposed approach in accomplishing these goals and to allow a quantitative comparison of results, we used the two metrics related to convergence and diversity described in [32] and we adopted the same testing procedures described there for sake of comparison. The computation of the values of such measures for a given test problem requires knowing its optimal front. For the used test problems, optimal fronts

have been derived using the specifications of the optimal solutions given in [32].

## The Convergence Metric

As described in [32], the convergence measure (denoted as  $\Theta$ ) gives an idea about the closeness of the obtained front to the true Pareto optimal front. It is measured as the average of the sum of minimum distances of each obtained non-dominated solution in the GBA to the true Pareto optimal front. The smaller the  $\Theta$  value, the better the convergence of the obtained front. Given an obtained Pareto front (GBA), the related value of  $\Theta$  is given by:

$$\Theta(GBA) = \frac{\sum_{i=1}^{|GBA|} d_i}{|GBA|} \quad \text{and } d_i = \min(d_{ij}), \quad j = 1..|Pareto\,optimal\,front|$$

Where  $d_{ij}$  is the Euclidean distance of the  $i^{th}$  member of GBA to the  $j^{th}$  member of the true optimal front.

## The Diversity Metric

The diversity metric [32] (denoted as  $\Delta$ ) measures how well the obtained front widely spread and uniformly distributed. Like the convergence metric, the smaller the  $\Delta$  metric value, the most uniformly spread out the set of non-dominated solutions. This metric is calculated according to the following equation given in [32]:

$$\Delta = \left( d_f + d_l + \sum_{i=1}^{|GBA|-1} |d_i - \bar{d}| \right) / (d_f + d_l + (|GBA| - 1)\bar{d})$$

where:

- $d_f$  and  $d_l$  are the Euclidean distances between the end solutions (boundary solutions) of the true Pareto front and the boundary solutions of the obtained front.
- $d_i$  is the  $i^{th}$  distance between two successive points of the obtained front in the objective space.
- $\bar{d}$  is the average of all  $(|GBA|-1)$  distances of  $d_i$ .

### 3.6.3 Parameter Settings

As mentioned earlier, the advantage of QPSO is that the contraction expansion parameter  $\beta$  is the only specific tunable parameter that needs to be set in addition to the two common parameters, the swarm size and maximum number of iterations.  $\beta$  is decreased linearly within the range  $[1.2 - 0.5]$ . Several runs of MOQPSO with different ranges of  $\beta$  have been performed on the employed test functions. Convergence and diversity values have been recorded for each test function and presented in Table 3.2. As it can be seen, results show that best results in terms of convergence and diversity for all test functions have been achieved within the range  $[1.2 - 0.5]$  which is in compliance with the settings established in the literature for  $\beta$  in the case of single objective optimization [105][104].

On the other hand, good quality Pareto fronts have been obtained with varying values of swarm size and maximum number of iterations parameters depending on the complexity of the test functions. For instance, the algorithm is set to perform 18000 fitness function evaluations for FON, 7500 fitness function evaluations for SCH, and 25000 fitness function evaluations for the ZDTs. For each test function, 30 independent runs were performed in order to collect the statistical results. The algorithm's parameter configurations for each test function are summarized in Table 3.3.

Range of $\beta$	Function name	Convergence	Diversity
[4 - 3.3]	FON	0.0835	0.4897
	SCH	0.0560	0.9210
	ZDT1	0.7805	0.7292
	ZDT2	0.8343	1
	ZDT3	0.3642	0.6884
	ZDT6	0.8525	1.1569
[2 - 1.2]	FON	0.0057	0.2459
	SCH	0.0033	0.2352
	ZDT1	0.1531	0.6329
	ZDT2	0.2985	0.6326
	ZDT3	0.1549	0.7613
	ZDT6	0.3365	0.8473
[0.5 - 0.1]	FON	0.0011	0.1875
	SCH	0.0031	0.1851
	ZDT1	0.0513	0.4456
	ZDT2	0.7027	N/A
	ZDT3	0.0498	0.9452
	ZDT6	0.5153	N/A
[1.2 - 0.5]	FON	0.0015	0.2594
	SCH	0.0030	0.1966
	ZDT1	0.0100	0.2361
	ZDT2	0.0140	0.2468
	ZDT3	0.0066	0.7941
	ZDT6	0.0289	0.8422

Tabel 3.2: Convergence and diversity values of the used test functions with different ranges of parameter  $\beta$

Test function	Population size	Number of Iterations	$\beta$	k
FON	300	60	1.2 - 0.5	10
SCH	150	50	1.2 - 0.5	10
ZDT1	100	250	1.2 - 0.5	10
ZDT2	100	250	1.2 - 0.5	10
ZDT3	100	250	1.2 - 0.5	10
ZDT6	100	250	1.2 - 0.5	10

Tabel 3.3: Parameter settings of MOQPSO for unconstrained test problems.



The number  $k$  of nearest neighbors in the proposed selection strategy has been set to 10. We decided to choose 10 as a value of  $k$  as we intend to maintain a balance between the effect of the sigma method and the crowding distance method on the proposed hybrid selection strategy. A small value of  $k$  (when  $k=1$ ) will promote the sigma method influence on the proposed selection strategy, while large value of  $k$  (when  $k=|GBA|$ ) will emphasize the crowding distance effect on the proposed selection scheme.

### 3.6.4 Experimental Results and Discussions

In Figure 3.3, we illustrate the Pareto fronts that have been obtained for each test function together with their true Pareto fronts. For qualitative assessment, the plots clearly show the ability of the proposed MOQPSO to achieve good quality Pareto fronts in terms of convergence and diversity simultaneously without using a mutation operator.

In order to quantitatively assess its performance, MOQPSO has been compared to the most well known algorithms for solving MOPs, namely NSGAI, SPEA and PAES. These algorithms represent the state-of-the-art. Results reported in [32] for unconstrained test problems have been used as the baseline to carry out this comparative study. The same setting and the same testing procedure as in [32] have been used for this purpose, i.e., the same number of function evaluations ( $\leq 25000$ ) have been carried out for the employed test functions. These test functions allow us to investigate different characteristics of the Pareto optimal front, such as convexity, non-convexity, connectivity, disconnectivity, sparsely distributed solutions near the Pareto optimal front and nonuniformity. The mean and variance of the convergence and diversity metrics have been recorded and presented in Tables 3.4 and 3.5 respectively to measure the performance of the algorithms.

In terms of convergence, as we can observe in Table 3.4, MOQPSO exhibits similar

results to NSGAI real coded and SPEA in SCH function for which PAES gives the best results.

For the second test function (FON), MOQPSO performs similarly to NSGAI real coded but presents better convergence values than PAES, SPEA and NSGAI binary coded.

For the third test function (ZDT1) and the fourth test function (ZDT2), MOQPSO's performance is inferior to NSGAI binary coded and SPEA, yet superior to NSGAI real coded and PAES.

MOQPSO outperforms the other algorithms in test functions 5 (ZDT3) and 6 (ZDT6).

It can be noticed that NSGAI binary coded is the only algorithm that shows comparable results to our proposed MOQPSO algorithm in terms of convergence to the true Pareto front.

As for diversity metric, it is clearly apparent from Figure 3.3 that an excellent spread among the set of obtained solutions has been achieved. This fact is corroborated by the results reported in Table 3.5. It is notable that MOQPSO outperforms the other algorithms in all cases, except for the sixth test function (ZDT6) where NSGAI real and binary coded present slightly better diversity values, without adopting any mutation mechanism. We attribute this to the dynamics of QPSO together with the hybrid selection mechanism which distributes the particles uniformly along the Pareto front.

It was not feasible to compare the running time of our algorithm with the published results in [32] because the published results were obtained using different platforms and different implementations. However, we are able to compare the per generation time complexities. Given  $N$  the population size and  $M$  the number of objectives, the time complexity of NSGAI is  $O(MN^2)$ , the time complexity of SPEA is  $O(MN^3)$  and the time complexity of PAES is  $O(MN^2)$  [32]. The time complexity of our algorithm is

$O(MN^2 \log N)$ . As it can be seen, the complexity of the proposed MOQPSO does not deviate too much compared to the other complexities. In addition, we can see that all these methods have in common a quadratic scaling with the population size  $N$ , and a linear scaling with the number of objective functions  $M$ . Our new method scales in the same way with  $M$ , and has an additional  $\log(N)$  scaling with  $N$  - a logarithmic term is not a big overhead and this is the price we pay for getting the improvements demonstrated in sections 3.3, 3.6.4, and 5.2.

Algorithms	Schaffer	Fonsesca	Zitzler1	Zitzler2	Zitzler3	Zitzler6
MOQPSO	0.0031	0.0015	0.0103	0.0136	0.0053	0.0389
	$4.56 * 10^{-9}$	$2.33 * 10^{-9}$	$3.40 * 10^{-7}$	$2.14 * 10^{-6}$	$1.05 * 10^{-7}$	$8.17 * 10^{-4}$
NSGAIIreal coded[32]	0.003391	0.001931	0.033482	0.072391	0.114500	0.296564
	0	0	0.004750	0.031689	0.007940	0.013135
NSGA II Binary coded[32]	0.002833	0.002571	0.000894	0.000824	0.043411	7.806798
	0.000001	0	0	0	0.000042	0.001667
SPEA [32]	0.003403	0.125692	0.001799	0.001339	0.047517	0.221138
	0	0.000038	0.000001	0	0.000047	0.000449
PAES [32]	0.001313	0.151263	0.082085	0.126276	0.023872	0.085469
	0.000003	0.000905	0.008679	0.036877	0.00001	0.006664

Tabel 3.4: Comparison with other MOEA: Convergence results. Mean (first row) and variance (second row)

Algorithms	Schaffer	Fonsesca	Zitzler1	Zitzler2	Zitzler3	Zitzler6
MOQPSO	0.2328	0.2372	0.2304	0.2156	0.5729	0.7908
	0.0001516	0.0004620	0.0009323	0.0007011	0.0018	0.0675
NSGAII real coded [32]	0.477899	0.378065	0.390307	0.430776	0.738540	0.668025
	0.003471	0.000639	0.001876	0.004721	0.019706	0.009923
NSGA II Binary coded[32]	0.449265	0.395131	0.463292	0.435112	0.575606	0.644477
	0.002062	0.001314	0.041622	0.024607	0.005078	0.035042
SPEA [32]	1.021110	0.792352	0.784525	0.755148	0.672938	0.849389
	0.004372	0.005546	0.004440	0.004521	0.003587	0.002713
PAES [32]	1.063288	1.162528	1.229794	1.165942	0.789920	1.153052
	0.002868	0.008945	0.004839	0.007682	0.001653	0.003916

Tabel 3.5: Comparison with other MOEA: Diversity results. Mean (first row) and variance (second row)

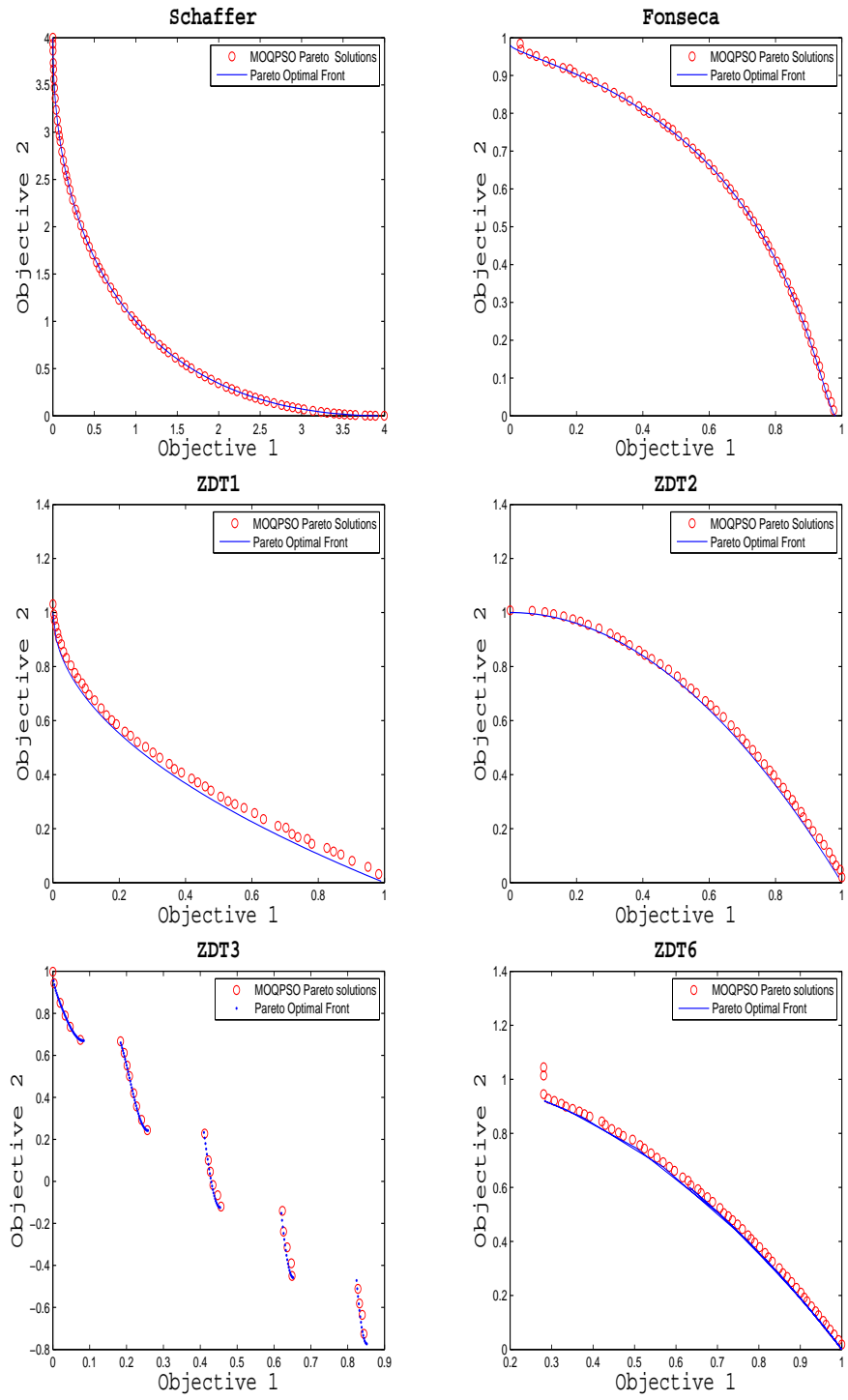


Figure 3.3: The true Pareto optimal and obtained fronts of the used test functions

### 3.7 Summary

In this chapter, we have built a novel framework for multi-objective optimization using QPSO (MOQPSO). We have also investigated the features related to the application of QPSO to handle multiple objectives. The main contribution is the proposal of a new leader selection strategy called the hybrid selection strategy that aims to achieve good convergence and diversity simultaneously without using any diversity preserving mechanism. It is a two level selection strategy that uses sigma values (to help convergence of each particle) and crowding distance information (to maintain diverse solutions) in order to select the suitable guide for each particle. MOQPSO has first generation time complexity of  $O(MN^2 \log N)$  which is only by a log factor larger than the well known multi-objective evolutionary algorithms, namely NSGAII, PAES and SPEA. From the experiments shown, we can say that the developed MOQPSO framework is a competitive algorithm to tackle MOPs and has demonstrated its effectiveness on unconstrained benchmark test problems.

As the MOQPSO framework was tested only on unconstrained test functions, an extension of it to handle constrained problems is required since most real-world problems include constraints. The following chapter will present such work.

---

### Constraint Handling in MOQPSO

---

This chapter presents an important development that extends the MOQPSO framework proposed in the previous chapter to enable it to handle constraints.<sup>1</sup> Two strategies to handle constraints are investigated. The first one is the death penalty strategy which discards infeasible solutions that are generated during the search process forcing the search process to explore only the feasible region. The second approach takes into account the infeasible solutions when computing the local attractors of particles and adopts a policy that achieves a balance between searching in infeasible and feasible regions.

#### 4.1 Constrained Multi-objective Optimization Problems (CMOPs)

Most real-world multi-objective optimization problems involve linear and/or non linear constraints that can be of equality or inequality type. Generally, constrained optimization problems are difficult to solve. This is because finding a solution that satisfies all

---

<sup>1</sup>A shorter version of the work in this chapter has been published in the following: Heyam Al-Baity, Souham Meshoul, and Ata Kaban. Constrained Multi-Objective Optimization using a Quantum Behaved Particle Swarm. The International Conference on Neural Information Processing (ICONIP 2012), Part III, LNCS 7665, pp. 456-464. Springer-Verlag Berlin Heidelberg, 2012.

constraints is not an easy task. The constraints split the search space into two regions depending on the feasibility of solutions. The feasible region encompasses all solutions satisfying all constraints. Hence, it contains all solutions of the problem. The infeasible region contains solutions that violate at least one of the constraints. Constraints can be categorized as hard – in which case they must be satisfied – or soft – in which case they may be satisfied to some extent [31].

There are a considerable number of methodologies found in the literature to solve constrained optimization problems with multiple objectives. One of these methodologies is to completely ignore infeasible solutions. Although it is simple, this approach may face difficulties in finding feasible solutions [31]. Penalty function methods are the most popular constraint handling techniques in evolutionary algorithms. In this method, penalty values are added to individuals violating the constraints [31]. Deb et al. [32] suggested a new idea to modify the definition of domination by turning it into constrained dominance of solutions by incorporating infeasible solutions during the search process.

Most of constraint handling methods for MOPs (CMOPs) proposed in the literature are used within evolutionary algorithms [26]. However, no in-depth study is available for handling constraints using swarm based algorithms like particle swarm optimization. In [22], Coello et al. proposed a simple scheme that has been used without a thorough investigation of how this impacts the search. The aim of this work is twofold. First, we study the ability of QPSO to deal with CMOPs and second we study the impact of discarding or keeping infeasible solutions during the search process. In chapter 3, we presented a new framework to extend QPSO to solve unconstrained MOPs which we called MOQPSO. In this chapter, we further extend this framework to handle constrained MOPs by investigating two strategies to deal with infeasible solutions generated during

the search process.

## 4.2 The Proposed MOQPSO for CMOPs

One of the main issues when solving constrained optimization problems is how to incorporate the infeasible solutions in the search process. In QPSO, as shown in equation (2.3) in section 2.2.2, the computation of a particle position requires calculating its attractor which is a function of the self best position of the particle and the global best position recorded within the whole swarm as described by equation (2.4) in section 2.2.2. Therefore, the level at which constraint handling can be considered when extending QPSO to solve constrained problems is when the local attractor of each particle has to be computed. This fact is behind the idea we propose in this work.

In our work, besides considering the use of the death penalty function as a first strategy to further extend MOQPSO to solve constrained problems, we also propose a second new strategy that considers the levels at which infeasible solutions may intervene when new positions are computed. In the following, we adopt the acronym CMOQPSO to refer to the proposed MOQPSO with constraint handling strategies.

### 4.2.1 CMOQPSO with the First Constraint Handling Strategy

Our first strategy consists simply in discarding infeasible solutions and using only the feasible ones throughout the search process. In this way, the whole swarm is forced to move within the feasible region. Therefore, only feasible solutions are used to update the local attractors of the particles, i.e., when an infeasible solution is generated, it is discarded until a feasible one is developed. The initialization process of particles positions is described in Algorithm 8.

In Algorithm 8, the function ‘check-constraint’ is checking if the current position sa-



---

**Algorithm 8** Initialize-Positions( )

---

```
1: Input: Population size, Problem dimension, Domain  $[min, max]$  in each dimension
2: for (each particle  $P_i$  ) do
3:   feasible = false
4:   repeat
5:     for (each dimension  $j$  ) do
6:        $x_{ij} = min_j + (max_j - min_j) * rand;$ 
7:     end for
8:     feasible = check-constraint( $X_i$ )
9:   until (feasible)
10: end for
11: Output : Particles positions  $X = [x_{ij}]$  for  $i=1..N$  and  $j=1..D$ 
```

---

tifies the problem constraints in which case it returns TRUE. The Algorithm 9 illustrates CMOQPSO with the first constraint handling strategy.

## 4.2.2 CMOQPSO with the Second Constraint Handling Strategy

The second strategy we propose to handle constraints couples the search in the feasible region with the search in the infeasible region. The aim is to take advantage of the infeasible solutions that could lead to promising areas in the feasible region.

As mentioned earlier, the local attractor is the means through which searching in the feasible and infeasible regions can be conducted. Hence, we suggest to explore both feasible and infeasible regions using new probabilistic selection rules of global best position and self best position. These selection rules are described as follows:

### Global Best Position Selection Rule

Global best infeasible solutions encountered during the search process are kept within an archive that we denote by Global Best Infeasible Archive (GBIA). The best infeasible solution is the one with the lowest constraint violation. In case of a tie, the best infeasible solution is the one with the better objective values because it is closer to the feasible

---

**Algorithm 9** Pseudocode of CMOQPSO with First Strategy

---

```
1: Input: MOP specification
2: N= population size
3: D= problem dimension
4:  $S^0$ = Initialize-Positions()
5:  $sbest_i$ = initialize self best position of particle  $P_i$  for  $i=1..N$ 
6: T= maximum number of iterations
7:  $\vec{F}_i$ = evaluate particle  $P_i$  for  $i=1..N$ 
8:  $GBA^0$ = initial set of non-dominated solutions
9:  $t = 1$ 
10:  $\beta^t = \beta_{max}$ 
11: repeat
12:   Compute mean best position using eq. (2.5)
13:   for (each particle  $P_i$ ) do
14:      $gbest = Select - leader(GBA^t, P_i)$ 
15:     feasible = false
16:     repeat
17:       for (each dimension j) do
18:          $p_{ij}^t = \text{Compute local attractor using eq. (2.4)}$ 
19:          $u_{ij} = rand(0, 1)$ 
20:         if  $rand(0, 1) > 0.5$  then
21:            $x_{ij}^{t+1} = p_{ij}^t + \beta^t \cdot |mbest_j^t - x_{ij}^t| \cdot \ln(1/u_{ij}^t)$  for  $j=1..D$ 
22:         else
23:            $x_{ij}^{t+1} = p_{ij}^t - \beta^t \cdot |mbest_j^t - x_{ij}^t| \cdot \ln(1/u_{ij}^t)$  for  $j=1..D$ 
24:         end if
25:       end for
26:       feasible = check-constraint( $X_i$ )
27:     until (feasible)
28:     Evaluate particle  $P_i$ 
29:     Update self best position
30:   end for
31:    $GBA^{t+1} = Update - Archive(GBA^t, S^t)$ 
32:    $\beta^{t+1} = \beta^t - \frac{(\beta_{max} - \beta_{min})}{T}$  {Decrease  $\beta$  linearly}
33:    $t = t + 1$ 
34: until ( $t > T$ )
35: Output :  $GBA$ 
```

---

region. In other words, the best infeasible solution is assessed from both the number of constraint violation and quality of objective functions. Given a particle for which a new position has to be computed, the global best solution for this particle is selected either from the Global best feasible archive (GBA) or the infeasible archive (GBIA) according to a probabilistic rule as follows:

---

**Algorithm 10** GLobal best selection rule

---

```
1:  $p = rand()$ ;
2: if  $p < P_G$  then
3:   select global best position from GBIA
4: else
5:   select global best position from GBA
6: end if
```

---

$P_G$  is the selection probability and can be set according to the importance we wish to devote to infeasible solutions. Selecting a leader from GBIA is straightforward. It consists of choosing the global leader randomly as all infeasible solutions in the GBIA have the same quality measure in terms of number of constraint violations. For GBA leader selection, we follow the same leader selection strategy we described in section 3.3, where the decision about which GBA member to choose as a leader for a given particle is made based on the closeness of each GBA member to the current particle in terms of sigma values and the extent to which the local area around the member is crowded.

## Updating Archives

Both archives, GBA and GBIA need to be updated after computing all particles' new positions. For a current particle, if a new infeasible solution is derived, its insertion in GBIA is considered. This new infeasible position enters the GBIA archive only if it has less constraint violation than any infeasible solution in GBIA. In this case, all GBIA contents with higher constraint violations have to be deleted from the archive. The new infeasible position is also included in GBIA in case it is equal to all GBIA solutions in terms of number of constraint violation and not dominated by any of the GBIA members. This update strategy is handled by the procedure 'Update-Archive-Infeasible'. On the other hand, GBA is updated as done with unconstrained MOQPSO. That is, if a new feasible solution is derived, it has to be checked against the GBA contents. The comparison here is based on the usual dominance concept. The new feasible position is entered into the GBA in a way that keeps GBA maintain only non-dominated solutions.

## Self Best Position Selection Rule

Basically, the strategy we followed in our work was to keep only one solution as a Self Best Feasible (SBF) point for each particle. In CMOQPSO, we keep track of the Self Best Infeasible solution (SBI) for each particle as well. The SBI solution is the one with lowest

constraint violations. When a new position has to be computed for a particle, the self best solution for this particle is selected either as the self best feasible (SBF) solution or the self best infeasible (SBI) solution according to a probabilistic rule as follows:

---

**Algorithm 11** Self best selection rule

---

```

1:  $p = rand()$ ;
2: if  $p < P_s$  then
3:   select SBI as the self best position
4: else
5:   select SBF as the self best position
6: end if

```

---

$P_s$  is the selection probability and can be set according to the importance we wish to devote to infeasible solutions. Finally, once self best and global best positions are determined for a particular particle, the local attractor can be computed as given in equation (2.4) in section 2.2.2.

## Updating the Self Best Position

Once the new position of a particle has been computed, its self best position has to be updated. If the new position is feasible, then SBF is updated based on the usual dominance relation. On the other hand, if the new position is infeasible, then SBI is updated with respect to the number of constraint violations.

## Modification of the Leader Selection Strategy

Because of the new proposed selection rules of global best and self best positions, the Select-Leader procedure described earlier in section 3.3 for unconstrained MOQPSO should be modified in a way to allow the combined search in feasible and infeasible regions. The ‘Modified-Select-Leader’ procedure includes the selection of a global best leader as well as a self best position according to the probabilistic selection rules that balance the search process in the two regions. The description of ‘Modified-Select-Leader’ procedure is as follows:

---

**Algorithm 12** Modified-Select-Leader (GBA, GBIA, SBF, SBI, P,  $P_G$ ,  $P_S$ )

---

```
1: Input: Current archive GBA, Current GBIA, SBF solution, SBI solution, Current particle P,  $P_G$ ,  $P_S$ 
2:  $p_1 = rand()$ ;
3: if  $p_1 < P_G$  then
4:   G= Select-randomly(GBIA)
5: else
6:   G= Select-Leader( GBA,P)
7: end if
8:  $p_2 = rand()$ ;
9: if  $p_2 < P_S$  then
10:  S=SBI
11: else
12:  S=SBF
13: end if
14: Output : G, S
```

---

According to the ‘Modified-Select-Leader’ procedure, given a particle for which a new position has to be computed, the global best position for this particle is selected either from the Global Best Archive (GBA) or the infeasible archive (GBIA) according to the probabilistic selection rule of *gbest*. In the same manner, the self best position for the given particle is selected either as the SBI or SBF based on the probabilistic selection rule of *sbest*. Finally, CMOQPSO with the second constraint handling strategy is described in Algorithm 13.

At the start of the algorithm, the N particles’ positions are initialized with uniformly distributed random numbers in the interval  $[0, 1]$ . These initial positions are then assigned to the corresponding self best positions. The pseudocode of the initialization process is given in Algorithm 14.

---

**Algorithm 13** Pseudocode of CMOQPSO with Second Strategy

---

```
1: Input: MOP specification
2: N= population size
3: D= problem dimension
4:  $S^0$ = Initialize-Positions()
5: T= maximum number of iterations
6:  $\vec{F}_i$ = evaluate Particle  $P_i$  for  $i=1..N$ 
7:  $SBF_i$ = initialize self best feasible position of particle  $P_i$  for  $i=1..N$ 
8:  $SBI_i$ = initialize self best infeasible position of particle  $P_i$  for  $i=1..N$ 
9:  $GBA^0$ = initial set of non-dominated feasible solutions
10:  $GBIA^0$ = initial set of best infeasible solutions
11:  $t = 1$ 
12:  $\beta^t = \beta_{max}$ 
13: repeat
14:   Compute mean best position using eq. (2.5)
15:   for (each particle  $P_i$  ) do
16:     [ $gbest, sbest$ ]= Modified-Select-Leader (GBA, GBIA, SBF, SBI, P,  $P_G$ ,  $P_S$ )
17:     for (each dimension  $j$  ) do
18:        $p_{ij}^t$  = Compute local attractor using eq. (2.4)
19:        $u_{ij} = rand(0, 1)$ 
20:       if  $rand(0, 1) > 0.5$  then
21:          $x_{ij}^{t+1} = p_{ij}^t + \beta^t \cdot |mbest_j^t - x_{ij}^t| \cdot \ln(1/u_{ij}^t)$  for  $j=1..D$ 
22:       else
23:          $x_{ij}^{t+1} = p_{ij}^t - \beta^t \cdot |mbest_j^t - x_{ij}^t| \cdot \ln(1/u_{ij}^t)$  for  $j=1..D$ 
24:       end if
25:     end for
26:     Evaluate particle  $P_i$ 
27:     feasible = check-constraint( $X_i$ )
28:     if (feasible) then
29:       Update self best feasible position (SBF)
30:     else
31:       Update self best infeasible position (SBI)
32:     end if
33:   end for
34:    $GBA^{t+1} = Update - Archive(GBA^t, S^t)$ 
35:    $GBIA^{t+1} = Update - Archive - infeasible(GBIA^t, S^t)$ 
36:    $\beta^{t+1} = \beta^t - \frac{(\beta_{max} - \beta_{min})}{T}$  {Decrease  $\beta$  linearly}
37:    $t = t + 1$ 
38: until ( $t \succ T$ )
39: Output :  $GBA$ 
```

---

---

**Algorithm 14** Initialize-Positions( )

---

```
1: Input: Population size, Problem dimension, Domain [ $min, max$ ] in each dimension
2: for (each particle  $P_i$  ) do
3:   for (each dimension  $j$  ) do
4:      $x_{ij} = min_j + (max_j - min_j) * rand;$ 
5:   end for
6:   [feasible, violations] = check-constraint( $X_i$ )
7: end for
8: Output : Particles positions  $X = [x_{ij}]$  for  $i=1..N$  and  $j=1..D$ 
```

---

In this case, the ‘check-constraint’ function returns whether the current solution is feasible or not and records the number of constraint violations.

### 4.3 Time Complexity Analysis of CMOQPSO for Constrained Problems

In the analysis of the constrained CMOQPSO algorithm, we consider the following parameters that influence the size of the problem:

- D: the problem dimension that is the number of decision variables
- M: the number of objective functions to be optimized
- N: the population size
- L: the archive size and
- R: the maximum number of trials required to produce a feasible solution

As described in the previous section 4.2, two variants of MOQPSO have been investigated to handle constraints according to two strategies. In the first strategy, infeasible solutions are discarded. The algorithm based on this strategy acts as MOQPSO for unconstrained problems. However, discarding infeasible solutions induces an extra computing time since the related positions are recomputed until feasible solutions are produced. The number of trials required to produce a feasible solution for a given particle is stochastic. Let  $R$  be the maximum number of trials. The strategy of discarding infeasible solutions impacts mainly the Update Particles’ positions task, for which the time complexity becomes  $O(RND)$ . The time complexity of the other tasks described earlier in section 3.5 remains the same as for unconstrained problems which is  $O(MN^2 \log N)$ . Therefore, the overall time complexity of the first strategy is  $O(MN^2 \log N + RND)$ .

In the second strategy, the main difference with MOQPSO for unconstrained problems consists of the modified leader selection strategy. In the worst case, this later acts as the proposed leader selection strategy described in section 3.3 the time complexity of which is at most  $O(MN^2 \log N)$  as explained in section 3.5.

As can be seen, the extra computational time required by the first strategy compared to the second strategy can be explained by the factor ( $RND$ ) due to discarding infeasible solution.

## 4.4 Experiments

In order to check the ability of the proposed CMOQPSO algorithm to optimize constrained problems, several benchmark test problems found in the specialized literature are employed. These test functions have been chosen because they have a variety of characteristics of the Pareto optimal front, such as convexity, non-convexity, connectivity and disconnectivity. They are broad and popular test functions for investigating the performance of multi-objective Pareto optimization methods. All the employed test functions are constrained minimization problems except for Kita test function and include two objectives. The definition of each of these test functions is presented next.

### Test Function 1

The first test function is called SRN [101]. The difficulty of this function lies in that the constraints exclude some part of the unconstrained Pareto front. It is defined as follows:



$$SRN : \left\{ \begin{array}{ll} \text{Minimize} & f_1(x) = (x_1 - 2)^2 + (x_2 - 1)^2 + 2 \\ \text{Minimize} & f_2(x) = 9x_1 - (x_2 - 1)^2 \\ \text{where:} & x_i \in [-20, 20], \quad \text{and} \quad i = 1, 2 \\ \text{subject to:} & g_1(x) = x_1^2 + x_2^2 \leq 225 \\ & g_2(x) = x_1 - 3x_2 \leq -10 \end{array} \right.$$

## Test Function 2

The CONSTR function [31] has a convex non-smooth Pareto optimal front. Similar to SRN, part of the unconstrained Pareto optimal front becomes infeasible due to the constraints. The two objective are:

$$CONSTR : \left\{ \begin{array}{ll} \text{Minimize} & f_1(x) = x_1 \\ \text{Minimize} & f_2(x) = (1 + x_2)/x_1 \\ \text{where:} & x_1 \in [0.1, 1.0] \quad \text{and} \quad x_2 \in [0, 5] \\ \text{subject to:} & g_1(x) = x_2 + 9x_1 \geq 6 \\ & g_2(x) = -x_2 + 9x_1 \geq 1 \end{array} \right.$$

## Test Function 3

The third test function is KITA [65]. It is defined as follows:

$$KITA : \left\{ \begin{array}{ll} \text{Maximize} & f_1(x, y) = -x^2 + 1 \\ \text{Maximize} & f_2(x, y) = 0.5x + y + 1 \\ \text{where:} & 0 \leq x, y \leq 7 \\ \text{subject to:} & x, y \geq 0 \\ & 0 \geq \frac{1}{6}x + y - \frac{13}{2} \\ & 0 \geq 0.5x + y - \frac{15}{2} \\ & 0 \geq 5x + y - 30 \end{array} \right.$$

## Test Function 4

The fourth test function is TNK [107]. It has a discontinuous Pareto optimal front. It is used to test the ability of the algorithm in finding a good spread of solutions along the Pareto front [33]. TNK is defined as follows:

$$TNK : \left\{ \begin{array}{ll} \text{Minimize} & f_1(x) = x_1 \\ \text{Minimize} & f_2(x) = x_2 \\ \text{where:} & x_i \in [0, \Pi], \quad \text{and} \quad i = 1, 2 \\ \text{subject to:} & g_1(x) = -x_1^2 - x_2^2 + 1 + 0.1 \cos(16 \arctan(x_1/x_2)) \leq 0 \\ & g_2(x) = (x_1 - 0.5)^2 + (x_2 - 0.5)^2 \leq 0.5 \end{array} \right.$$

## Test Function 5

The MOBES function is proposed by Binh et al. [15]. It has a non-convex feasible region. It is defined as follows:

$$MOBES : \left\{ \begin{array}{ll} \text{Minimize} & f_1(x_1, x_2) = -4x_1^2 + 4x_2^2 \\ \text{Minimize} & f_2(x_1, x_2) = (x_1 - 5)^2 + (x_2 - 5)^2 \\ \text{where:} & -15 \leq x_i \leq 30 \quad \forall i = 1, 2 \\ \text{subject to:} & (x_1 - 5)^2 + x_2^2 - 25 \leq 0 \\ & -(x_1 - 8)^2 - (x_2 + 3)^2 + 7.7 \leq 0 \end{array} \right.$$

#### 4.4.1 Experimental Set up

In order to assess the performance of CMOQPSO using the two constraint handling strategies described in section 4.2, 10 independent runs for each test function in each constraint handling strategy have been conducted. In all experiments, we follow the same parameter settings of  $\beta$  and  $k$  we used for unconstrained MOQPSO in the previous chapter. The contraction expansion parameter  $\beta$  has been decreased linearly within the range [1.2 - 0.5] which is in compliance with the settings reported in the literature [105][104] as it shows good results in terms of convergence and diversity. The number  $k$  of neighbors in the selection of the global feasible leader has been set to 10 to maintain a balance between the effect of the sigma method and the crowding distance method on the proposed hybrid selection strategy. The selection probability  $P_G$  is set to 0.5 in order to maintain a balance between the search in the feasible and the infeasible regions and  $P_S$  is set to 0.3 to favor the feasible self best positions.

Now we are going to describe how we set the number of particles and the number of iterations. We started by running the algorithm with a small number of particles (60) and a small number of iterations (100) and we plotted the obtained Pareto front. We then continued repeating the process with an increase in the number of particles (100, 120, 150) and an increase in the number of iterations ( 200, 300, 400, 500 ) until we reached a

good quality Pareto front in terms of convergence and diversity for all the test functions. Finally, we got good quality solutions with the following setting: the number of particles is set to 150 for all test functions and the maximum number of iterations has been set to 700 for KITA function and 500 for the remaining functions. The parameter configurations for CMOQPSO with the first and the second constraint handling strategies for each test function are summarized in Table 4.1.

Test function	Population size	Number of Iterations	$\beta$	k
CONSTR	150	500	1.2 - 0.5	10
MOBES	150	500	1.2 - 0.5	10
SRN	150	500	1.2 - 0.5	10
KITA	150	700	1.2 - 0.5	10
TNK	150	500	1.2 - 0.5	10

Tabel 4.1: Parameter settings of CMOQPSO with first and second constraint handling strategies for the constraint test functions.

#### 4.4.2 Performance Measures

In order to evaluate the performance of both strategies in terms of convergence and diversity of the obtained fronts, the two metrics that have been used for measuring the performance of unconstrained MOQPSO, namely the convergence and diversity metrics described in section 3.6.2 have been also used for CMOQPSO.

#### 4.4.3 Experimental Results and Discussions

Figures 4.1 and 4.2 show the obtained Pareto fronts along with the true Pareto optimal fronts using the first and the second constraint handling strategies for SRN, MOBES, KITA, TNK, and CONSTR test functions. At a first glance, we can see that both strategies are similar in achieving very good convergence and diversity; although it can be noticed that the first strategy exhibits a slightly better performance when compared with the second strategy. To corroborate this fact, quantitatively speaking, ten runs in each strategy for each test function have been conducted. Statistical results have been gathe-

red in Table 4.2. This table presents the convergence and the diversity metric values for the two strategies together with the p-value as obtained from the Wilcoxon rank sum test. This latter is a nonparametric test used to compare the median of two samples. The convergence metric computes the average distance between the obtained front and the true Pareto front. The smaller the value of this metric, the better the convergence of the obtained Pareto front. The diversity metric measures how uniformly are spread the obtained non-dominated solutions. The smaller the value of the diversity metric, the most widely and uniformly spread the set of non-dominated solutions. The p-value is a probability that indicates if there is a statistically significant difference between the two samples. If the p-value is less than or equal to the significance level 0.05, then there is a difference between the results of the two algorithms. Otherwise, there is no difference detected between the performance of the compared algorithms. The values of the standard deviation show the robustness and the high quality of the found solutions in both strategies.

Regarding the two investigated strategies, it is apparent from Table 4.2 that discarding infeasible solutions during the search process (first strategy) slightly outperforms the second strategy with respect to convergence in all cases and diversity in most of the cases. The first strategy has led to the best results from both convergence and diversity points of view in case of SRN, KITA, and MOBES functions whereas the second strategy is more efficient in terms of diversity in case of TNK and CONSTR functions only. The superiority of the first strategy over the second is because the second strategy is mainly based on the number of constraint violations when evaluating the infeasible solutions. This seems not enough when handling constraints through involving infeasible solutions in the search process. Another criterion should be focused on, together with the number of constraint violations, which is the distance measure of the infeasible solutions to the feasible region

boundary when evaluating infeasible solutions. By adding this new criterion, the second constraint handling strategy might be improved.

However, the first strategy requires more computational time than the second strategy. This can be explained by the fact that with the death penalty function, as long as an infeasible solution is generated, a new one is recomputed until a feasible solution is obtained.

The Wilcoxon rank sum test [34] with  $\alpha = 0.05$  is used to assess the difference between the two constraint handling strategies. We can see from the p-values shown in Table 4.2 that there is a high significant statistical difference between the two compared strategies in most of the cases except on the diversity metric for KITA and TNK functions.

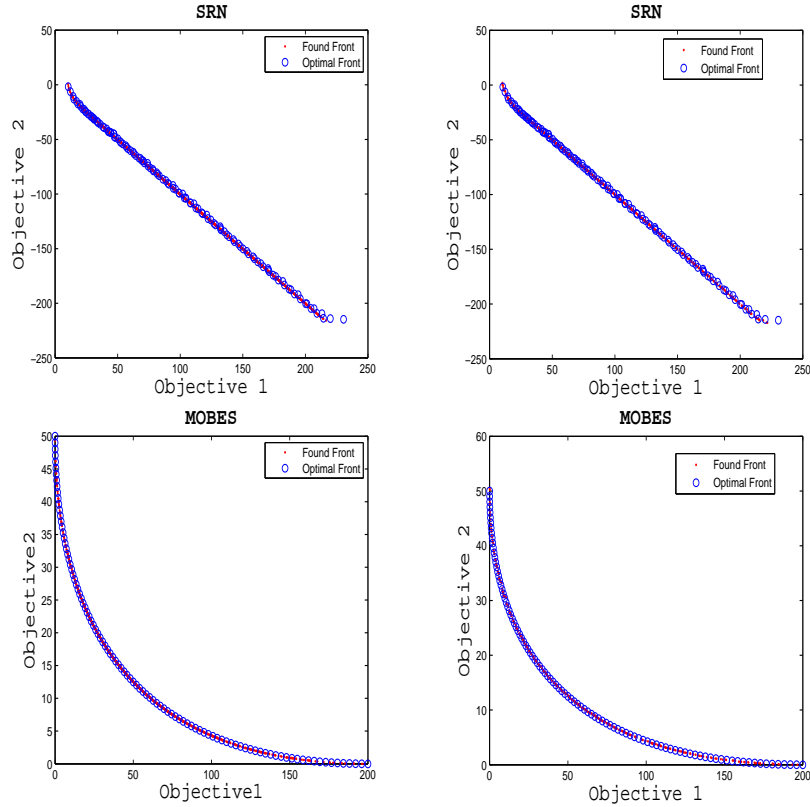
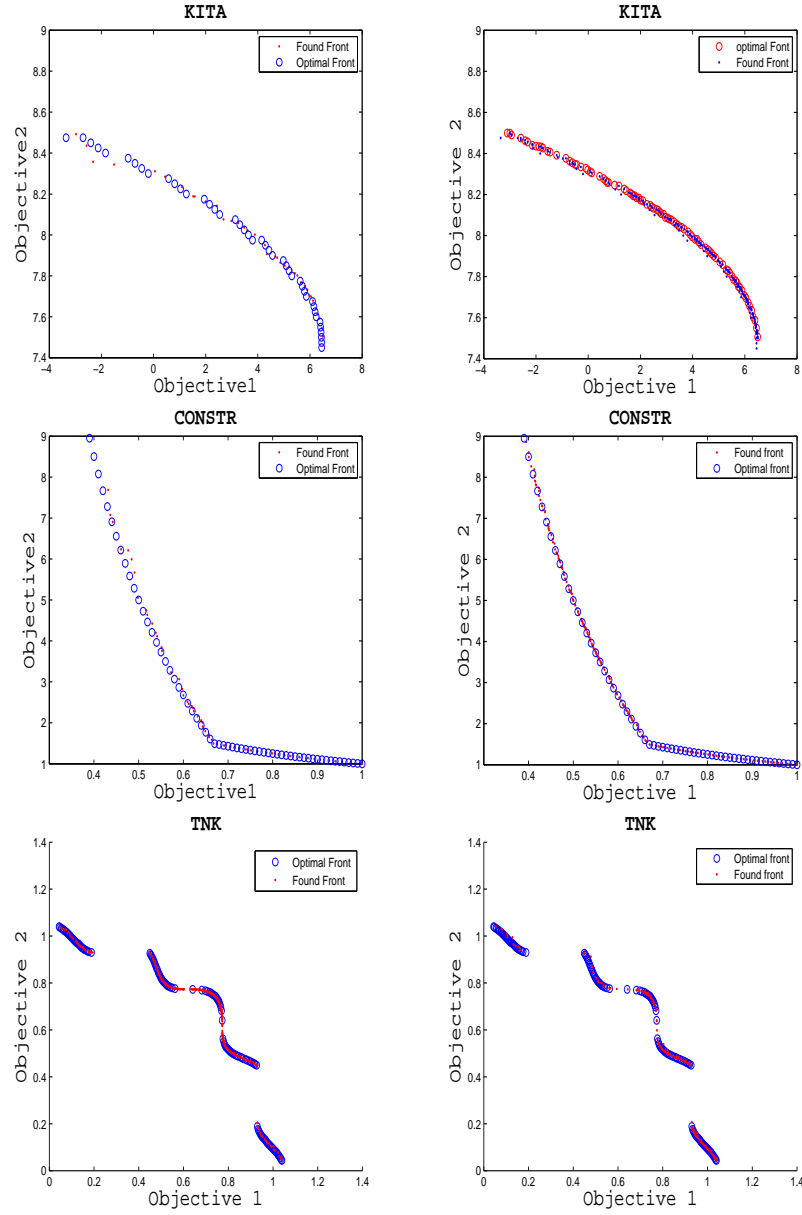


Figure 4.1: The true Pareto optimal and the obtained fronts of SRN and MOBES test functions using the first strategy (left column) and the second strategy (right column).



Figur 4.2: The true Pareto optimal and the obtained fronts of KITA, CONSTR, and TNK test functions using the first strategy (left column) and the second strategy (right column).

Tabel 4.2: Convergence and diversity values of the used test functions for the first and the second constraint handling strategies

Test Problem	Statistics	Convergence			Diversity		
		First	Second	P-value	First	Second	P-value
SRN	Best	<b>0.2916</b>	0.3060	$1.826 * 10^{-4}$	<b>0.2651</b>	0.3407	$1.816 * 10^{-4}$
	Worst	<b>0.3072</b>	0.3351		<b>0.2987</b>	0.3981	
	Average	<b>0.2986</b>	0.3198		<b>0.2878</b>	0.3789	
	Median	<b>0.3000</b>	0.3208		<b>0.2883</b>	0.3828	
	Std.	<b>0.0051</b>	0.0085		<b>0.0113</b>	0.0177	
CONSTR	Best	<b>0.0049</b>	0.0130	$1.726 * 10^{-4}$	0.5702	<b>0.3984</b>	$1.776 * 10^{-4}$
	Worst	<b>0.0054</b>	0.0184		0.7188	<b>0.5486</b>	
	Average	<b>0.0051</b>	0.0159		0.6537	<b>0.4444</b>	
	Median	<b>0.0051</b>	0.0158		0.6611	<b>0.4431</b>	
	Std.	<b>0.0001</b>	0.0016		0.0543	<b>0.0439</b>	
KITA	Best	<b>0.0555</b>	0.0891	0.0017	0.2572	0.2139	0.9097
	Worst	<b>0.1127</b>	0.2483		0.7021	0.8577	
	Average	<b>0.0719</b>	0.1399		0.4580	0.5017	
	Median	<b>0.0606</b>	0.1317		0.4302	0.4707	
	Std.	<b>0.0210</b>	0.0495		0.1313	0.2332	
MOBES	Best	<b>0.2535</b>	0.3059	$1.736 * 10^{-4}$	<b>0.3174</b>	0.3326	$1.726 * 10^{-4}$
	Worst	<b>0.2651</b>	0.3449		<b>0.2788</b>	0.4764	
	Average	<b>0.2582</b>	0.3281		<b>0.3017</b>	0.3899	
	Median	<b>0.2575</b>	0.3305		<b>0.3025</b>	0.3884	
	Std.	<b>0.0040</b>	0.0138		<b>0.0147</b>	0.0389	
TNK	Best	<b>0.0081</b>	0.0084	$4.288 * 10^{-4}$	0.4319	0.3751	0.850
	Worst	<b>0.0094</b>	0.0107		0.5187	0.5230	
	Average	<b>0.0087</b>	0.0097		0.4884	0.4690	
	Median	<b>0.0089</b>	0.0096		0.4995	0.4749	
	Std.	<b>0.0005</b>	0.0006		0.0305	0.0404	



#### 4.4.4 Study of the Effect of the Probabilistic Selection Rules of *gbest* and *sbest* positions

In this section, we investigate the impact of the probabilistic selection values ( $P_G$  and  $P_S$ ) on the performance of CMOQPSO with the second constraint handling strategy. Different combinations of  $P_G$  and  $P_S$  have been set and tested on each test function. The statistical results of convergence and diversity metrics were collected after conducting 5 runs for each test function in each combination of ( $P_G$  and  $P_S$ ) and presented in Table 4.4 until Table 4.9. The infeasible self best and global best combinations we used in our experiments are presented in Table 4.3.

% of incorporation of infeasible <i>sbest</i>	% of incorporation of infeasible <i>gbest</i>
30%	30%, 50%, and 70%
50%	30%, 50%, and 70%
70%	30%, 50%, and 70%

Tabel 4.3: The different combinations of infeasible *sbest* and *gbest* used in this study

In general, it can be seen statistically that this study did not find a significant difference between the results under the different combinations of infeasible self best and infeasible global best.

With respect to the convergence metric, the 30% favouring infeasible global best always has better performance with the different percentages of infeasible self best. Moreover, the combination of incorporating 30% of infeasible self best and 30% of infeasible global best shows better convergence values. On the other hand, the algorithm behaves very similar from diversity point of view.

As a result, we can say that CMOQPSO performs better when infeasible solutions of self best or global best are less employed in the search process. Therefore, we tested the algorithm behaviour when using only feasible self best positions, i.e., 0% of using infeasible self best solutions. From Tables 4.10 and 4.11, we can observe that CMOQPSO gained a slight improvement in its performance in terms of convergence and diversity. We can attribute this fact to the constraint strategy technique we propose that is based on the number of constraint violations only when dealing with the infeasible solutions, which seems not enough. In order to get the advantage of incorporating the infeasible solutions during the search process, we may need to improve the second constraint strategy by adding the distance of infeasible solutions to the feasible region boundary as another criterion when evaluating infeasible solutions.

<b>30% favouring infeasible <i>gbest</i>(Convergence)</b>						
		Max	Min	Mean	Median	Std
<b>30% favouring infeasible <i>sbest</i></b>	CONSTR	0.0176	0.0164	0.0170	0.0172	0.0005
	MOBES	0.3172	0.2784	0.2945	0.2913	0.0162
	SRN	0.3049	0.2843	0.2952	0.2991	0.0087
	KITA	0.2158	0.0671	0.1228	0.1098	0.0615
	TNK	0.0103	0.0089	0.0096	0.0096	0.0006
	<b>50% favouring infeasible <i>gbest</i></b>					
	CONSTR	0.0184	0.0130	0.0159	0.0158	0.0020
	MOBES	0.3449	0.3094	0.3334	0.3395	0.0144
	SRN	0.3351	0.3157	0.3230	0.3208	0.0072
	KITA	0.2483	0.0891	0.1580	0.1541	0.0597
	TNK	0.0116	0.0090	0.0107	0.0114	0.0012
	<b>70% favouring infeasible <i>gbest</i></b>					
	CONSTR	0.0250	0.0204	0.0235	0.0241	0.0019
	MOBES	0.3426	0.2981	0.3197	0.3204	0.0188
	SRN	0.3263	0.3122	0.3180	0.3138	0.0069
	KITA	0.2284	0.0796	0.1516	0.1377	0.0696
	TNK	0.0142	0.0101	0.0115	0.0111	0.0016

Tabel 4.4: Statistical results of convergence metric with 30% favouring infeasible *sbest*

<b>30% favouring infeasible <i>gbest</i> (Diversity)</b>						
		Max	Min	Mean	Median	Std
<b>30% favouring infeasible <i>sbest</i></b>	CONSTR	0.3666	0.3047	0.3317	0.3357	0.0242
	MOBES	0.3366	0.3018	0.3213	0.3266	0.0169
	SRN	0.3385	0.3079	0.3249	0.3265	0.0121
	KITA	1.1130	0.7386	0.9289	1.0007	0.1640
	TNK	0.5175	0.4742	0.4972	0.5004	0.0162
	<b>50% favouring infeasible <i>gbest</i></b>					
	CONSTR	0.5486	0.3984	0.4598	0.4478	0.0554
	MOBES	0.4764	0.3590	0.4013	0.3926	0.0456
	SRN	0.3932	0.3638	0.3801	0.3870	0.0137
	KITA	0.8577	0.2405	0.5408	0.4983	0.2330
	TNK	0.5188	0.4666	0.4935	0.4941	0.0185
	<b>70% favouring infeasible <i>gbest</i></b>					
	CONSTR	0.3507	0.3341	0.3413	0.3375	0.0073
	MOBES	0.4839	0.3906	0.4321	0.4217	0.0370
	SRN	0.4100	0.3360	0.3566	0.3450	0.0302
	KITA	1.1239	0.7560	0.9794	1.0889	0.1693
	TNK	0.5113	0.4542	0.4744	0.4688	0.0216

Tabel 4.5: Statistical results of diversity metric with 30% favouring infeasible *sbest*

<b>30% favouring infeasible <i>gbest</i>(Convergence)</b>						
		Max	Min	Mean	Median	Std
<b>50% favouring infeasible <i>sbest</i></b>	CONSTR	0.0191	0.0164	0.0177	0.0178	0.0011
	MOBES	0.3252	0.2736	0.3065	0.3148	0.0214
	SRN	0.3203	0.2954	0.3073	0.3093	0.0093
	KITA	0.0984	0.0738	0.0887	0.0946	0.0105
	TNK	0.0135	0.0089	0.0111	0.0109	0.0016
	<b>50% favouring infeasible <i>gbest</i></b>					
	CONSTR	0.0221	0.0185	0.0207	0.0210	0.0014
	MOBES	0.3414	0.2984	0.3138	0.3050	0.0177
	SRN	0.3411	0.3118	0.3208	0.3183	0.0117
	KITA	0.1399	0.0707	0.0935	0.0866	0.0277
	TNK	0.0129	0.0101	0.0117	0.0121	0.0011
	<b>70% favouring infeasible <i>gbest</i></b>					
	CONSTR	0.0255	0.0221	0.0242	0.0248	0.0013
	MOBES	0.3464	0.3062	0.3270	0.3298	0.0159
	SRN	0.3366	0.3200	0.3285	0.3297	0.0066
	KITA	0.1842	0.0969	0.1412	0.1297	0.0393
	TNK	0.0137	0.0099	0.0119	0.0120	0.0017

Tabel 4.6: Statistical results of convergence metric with 50% favouring infeasible *sbest*

<b>30% favouring infeasible <i>gbest</i> (Diversity)</b>						
		Max	Min	Mean	Median	Std
<b>50% favouring infeasible <i>sbest</i></b>	CONSTR	0.3479	0.2992	0.3266	0.3333	0.0214
	MOBES	0.3790	0.3139	0.3508	0.3587	0.0266
	SRN	0.3605	0.3126	0.3341	0.3344	0.0172
	KITA	1.0097	0.7390	0.8309	0.7947	0.1074
	TNK	0.5404	0.4442	0.4827	0.4884	0.0396
	<b>50% favouring infeasible <i>gbest</i></b>					
	CONSTR	0.3689	0.3096	0.3452	0.3564	0.0267
	MOBES	0.4581	0.3531	0.3964	0.3816	0.0414
	SRN	0.3608	0.3209	0.3377	0.3331	0.0164
	KITA	0.9929	0.7320	0.8392	0.8360	0.1000
	TNK	0.5779	0.4467	0.5207	0.5255	0.0477
	<b>70% favouring infeasible <i>gbest</i></b>					
	CONSTR	0.4238	0.2895	0.3488	0.3402	0.0487
	MOBES	0.4678	0.4052	0.4407	0.4409	0.0252
	SRN	0.3867	0.3301	0.3603	0.3686	0.0221
	KITA	1.0826	0.6694	0.9155	0.9625	0.1528
	TNK	0.5757	0.4070	0.5116	0.5455	0.0708

Tabel 4.7: Statistical results of diversity metric with 50% favouring infeasible *sbest*

30% favouring infeasible <i>gbest</i> (Convergence)						
		Max	Min	Mean	Median	Std
70% favouring infeasible <i>sbest</i>	CONSTR	0.0222	0.0182	0.0203	0.0209	0.0016
	MOBES	0.3437	0.2916	0.3220	0.3263	0.0193
	SRN	0.3165	0.3047	0.3115	0.3137	0.0057
	KITA	0.1726	0.0764	0.1193	0.1152	0.0346
	TNK	0.0116	0.0108	0.0112	0.0112	0.0003
	50% favouring infeasible <i>gbest</i>					
	CONSTR	0.0258	0.0215	0.0239	0.0238	0.0017
	MOBES	0.3549	0.3270	0.3397	0.3382	0.0103
	SRN	0.3499	0.3157	0.3293	0.3304	0.0137
	KITA	0.2337	0.0933	0.1344	0.1119	0.0568
	TNK	0.0130	0.0097	0.0118	0.0128	0.0015
	70% favouring infeasible <i>gbest</i>					
	CONSTR	0.0304	0.0223	0.0261	0.0257	0.0029
	MOBES	0.3709	0.3369	0.3564	0.3612	0.0140
	SRN	0.3558	0.3275	0.3391	0.3402	0.0113
	KITA	0.2588	0.1511	0.1857	0.1682	0.0429
	TNK	0.0147	0.0104	0.0123	0.0121	0.0015

Tabel 4.8: Statistical results of convergence metric with 70% favouring infeasible *sbest*

30% favouring infeasible <i>gbest</i> (Diversity)						
		Max	Min	Mean	Median	Std
70% favouring infeasible <i>sbest</i>	CONSTR	0.3511	0.3078	0.3262	0.3181	0.0204
	MOBES	0.4114	0.3625	0.3859	0.3935	0.0207
	SRN	0.3654	0.3215	0.3412	0.3358	0.0164
	KITA	1.0160	0.7119	0.8686	0.8872	0.1141
	TNK	0.4921	0.4358	0.4680	0.4739	0.0240
	50% favouring infeasible <i>gbest</i>					
	CONSTR	0.3740	0.2987	0.3377	0.3509	0.0319
	MOBES	0.4478	0.3945	0.4220	0.4218	0.0191
	SRN	0.3906	0.3357	0.3585	0.3522	0.0221
	KITA	1.0869	0.6563	0.8829	0.9294	0.1681
	TNK	0.5140	0.4672	0.4903	0.4903	0.0185
	70% favouring infeasible <i>gbest</i>					
	CONSTR	0.3654	0.3325	0.3482	0.3492	0.0118
	MOBES	0.5261	0.4161	0.4580	0.4526	0.0413
	SRN	0.4097	0.3333	0.3670	0.3679	0.0283
	KITA	1.0718	0.8808	1.0045	1.0012	0.0773
	TNK	0.5201	0.4696	0.4971	0.5085	0.0239

Tabel 4.9: Statistical results of diversity metric with 70% favouring infeasible *sbest*



<b>30% favouring infeasible <i>gbest</i>(Convergence)</b>						
		Max	Min	Mean	Median	Std
<b>0% favouring infeasible <i>sbest</i></b>	CONSTR	0.0153	0.0132	0.0143	0.0142	0.0008
	MOBES	0.3065	0.2856	0.2994	0.3027	0.0087
	SRN	0.2989	0.2919	0.2955	0.2947	0.0033
	KITA	0.1182	0.0706	0.0841	0.0797	0.0197
	TNK	0.0108	0.0089	0.0096	0.0094	0.0008
	<b>50% favouring infeasible <i>gbest</i></b>					
	CONSTR	0.0201	0.0183	0.0189	0.0188	0.0007
	MOBES	0.3070	0.2966	0.3033	0.3059	0.0047
	SRN	0.3121	0.2846	0.2971	0.2963	0.0098
	KITA	0.1147	0.0878	0.0992	0.0923	0.0127
	TNK	0.0121	0.0077	0.0097	0.0096	0.0016
	<b>70% favouring infeasible <i>gbest</i></b>					
	CONSTR	0.0246	0.0204	0.0221	0.0213	0.0017
	MOBES	0.3405	0.3141	0.3260	0.3234	0.0114
	SRN	0.3291	0.3093	0.3168	0.3164	0.0079
	KITA	0.1427	0.0954	0.1200	0.1241	0.0225
	TNK	0.0124	0.0088	0.0109	0.0118	0.0018

Tabel 4.10: Statistical results of convergence metric with 0% favouring infeasible *sbest*

30% favouring infeasible <i>gbest</i> (Diversity)						
		Max	Min	Mean	Median	Std
0% favouring infeasible <i>sbest</i>	CONSTR	0.3596	0.3213	0.3316	0.3248	0.0159
	MOBES	0.3049	0.2347	0.2768	0.2838	0.0258
	SRN	0.2972	0.2642	0.2801	0.2865	0.0151
	KITA	1.1295	0.7028	0.8263	0.7755	0.1742
	TNK	0.5115	0.4516	0.4772	0.4765	0.0218
	50% favouring infeasible <i>gbest</i>					
	CONSTR	0.3472	0.2900	0.3236	0.3246	0.0212
	MOBES	0.3623	0.2912	0.3217	0.3189	0.0276
	SRN	0.3196	0.2978	0.3093	0.3097	0.0081
	KITA	1.0750	0.8080	0.8800	0.8265	0.1132
	TNK	0.5357	0.4608	0.5026	0.5185	0.0338
	70% favouring infeasible <i>gbest</i>					
	CONSTR	0.3359	0.2753	0.3002	0.2947	0.0269
	MOBES	0.4607	0.3040	0.3485	0.3237	0.0647
	SRN	0.3533	0.3228	0.3385	0.3434	0.0143
	KITA	0.9797	0.7082	0.8161	0.8111	0.1038
	TNK	0.5214	0.4290	0.4769	0.4829	0.0335

Tabel 4.11: Statistical results of diversity metric with 0% favouring infeasible *sbest*

## 4.5 Summary

In this chapter, we further developed and investigated the use of QPSO to handle constrained multi-objective optimization problems. Two strategies to deal with infeasible solutions have been studied that consist of discarding versus keeping infeasible solutions. In both cases, the extended QPSO has been successfully applied to CMOPs. The first strategy has been found to achieve the best results in terms of convergence and diversity in most of the cases. However, this comes at the expense of an additional  $O(RND)$  computation

time per generation in comparison to the second constraint strategy. The second constraint strategy has the same scaling as the MOQPSO for unconstrained problems from the previous chapter.

---

# An Extensive Empirical Comparison of Different Selection Strategies for Constrained and Unconstrained Problems

---

This chapter provides an extensive study of the potential of MOQPSO under several leader selection strategies on unconstrained and constrained test problems.<sup>1</sup> Besides, a comparative study of MOQPSO with MOPSO is performed.

We present a thorough comparison of various guide selection methods for global best position, namely the proposed hybrid strategy, the crowding distance method, the sigma method, and the random selection method in order to study the effect of combining the sigma method and the crowding distance method in the proposed hybrid strategy and to investigate which of these methods is likely to give the best performance in terms of convergence, diversity and computational time. Furthermore, we replaced the QPSO in

---

<sup>1</sup>The content of this and the next chapter has been submitted to the following: Heyam Al-Baity, Souham Meshoul, and Ata Kaban. Swarm Based Multi-Objective Optimization with Quantum Behaved Particles. International Journal of Bio-Inspired Computation (IJBIC), submitted, 2014.

our proposed MOQPSO with PSO in order to compare the behavior of both algorithms in the same environment and under the same selection and archiving strategies.

A thorough experimental study has been conducted to gauge the behavior of MOQPSO under different scenarios regarding the leader selection methods for unconstrained and constrained test problems. In order to quantitatively assess the performance of MOQPSO, three important questions have been addressed:

1. What is the impact of the leader selection strategies on the algorithm performance in terms of convergence, diversity, and run time?
2. What is the impact of the constraint handling strategies on the performance of MOQPSO with different leader selection methods?
3. Does QPSO behave better than PSO in multi-objective context and to what extent?

In the following, we will describe the experiments designed to answer each of above questions along with a discussion of the obtained results.

## 5.1 Experiments

All experiments are evaluated using the same benchmark test problems employed in chapters 3 and 4 for the same reasons explained in section 3.6.1 for unconstrained problems and section 4.4 for constrained problems. For each test problem in each method, 30 independent runs are performed.

### 5.1.1 Experiment1: Impact of Leader Selection Strategies

This experiment has been conducted to study the impact of the proposed hybrid leader selection strategy on unconstrained and constrained MOQPSO compared to other selection strategies. The idea is, this strategy selects the leader for a particular particle as the less

crowded solution among the  $k$  nearest GBA members in terms of sigma values. When  $k$  is too small ( $k = 1$ ), it tends to behave like the sigma method and when  $k$  is too large ( $k = |GBA|$ ), it tends to behave like the crowding distance based selection method. We recall that the objective behind the proposed design of the hybrid strategy is to achieve a good convergence while preserving a good spread of solutions along the front. In this experiment, MOQPSO has been run with the four different selection strategies, namely

1. The proposed hybrid selection strategy
2. The crowding distance based selection strategy
3. The sigma based selection strategy
4. The random selection strategy

All experiments are conducted using the unbounded archive strategy. In this experiment, we followed the same parameter settings used in the previous chapters for unconstrained and constrained test problems as explained in sections 3.6.3 and 4.4.1. These settings are summarized in Table 5.1 for unconstrained problems and Table 5.2 for constrained problems with  $\beta$  value ranging in  $[1.2 - 0.5]$  and  $k = 10$ . The selection probability  $P_G$  is set to 0.5 and  $P_S$  is set to 0.3.

The values of the convergence and the diversity metrics of the obtained fronts are illustrated by statistical box plots as shown in Tables 5.3, 5.4, 5.5, and 5.6 respectively. The meaning of those metrics are explained in section 4.4.3, page 104. In order to claim that the statistical results of the proposed hybrid selection method are significantly different from the other considered methods, Friedman test has been conducted. Friedman test is a non-parametric statistical test used for detecting the difference between several related samples [34][28]. The Friedman test is followed by a multicomparison test [28] in order

to provide graphs that show the difference between each pair of methods. The p-values obtained from the Friedman test at the significance level  $\alpha = 0.05$  are given with the box plots in Table 5.3 - Table 5.6. The p-value is a probability that indicates if there is a statistically significant difference between the related samples. If the p-value is less than or equal to the significance level 0.05 then there is at least one of the samples differs from the rest. Otherwise, there is no difference detected between the performance of the compared methods. The graphs related to multicomparison tests are shown in Figure 5.1 for the convergence metric and in Figure 5.2 for the diversity metric.

Test function	Population size	Number of Iterations
FON	300	60
SCH	150	50
ZDT1	100	250
ZDT2	100	250
ZDT3	100	250
ZDT6	100	250

Tabel 5.1: Parameter settings of MOQPSO for experiments to compare different selection methods on unconstrained test problems.

Test function	Population size	Number of Iterations
CONSTR	150	100
MOBES	150	150
SRN	150	150
KITA	250	100
TNK	250	100

Tabel 5.2: Parameter settings of CMOQPSO with the first and the second constraint handling strategies for experiments to compare different selection methods.

The results related to the convergence metric in Tables 5.3 and 5.4 show that the hybrid strategy exhibits better performance than the random selection strategy and the crowding distance strategy for all unconstrained test problems. According to the p-values in Tables 5.3 and 5.4 and the multicomparison tests in Figure 5.1, there is a significant difference between the hybrid strategy and the random strategy in all test functions except for SCH function. On the other hand, the hybrid strategy achieves significantly different results on SCH and FON compared to the crowding method and no significant difference on ZDTs functions. Compared to the sigma method, the hybrid method achieves better results in FON, SCH, and ZDT2 functions with a significant difference on ZDT2. The sigma method has a slightly better performance on the remaining functions ZDT1, ZDT3, and ZDT6 with no significant difference with the hybrid selection strategy. However, the sigma method performed quite badly when solving ZDT2 test function. The algorithm with sigma method finds it hard to converge to the true Pareto front due to the selection pressure problem that the sigma method suffers from, which may cause premature convergence. On the other hand, the hybrid method is the one which performs the best for this test problem. In general, as expected, the hybrid method performs better than the crowding and random selection methods and competes with the sigma method even achieves better convergence results than the sigma method in some cases.

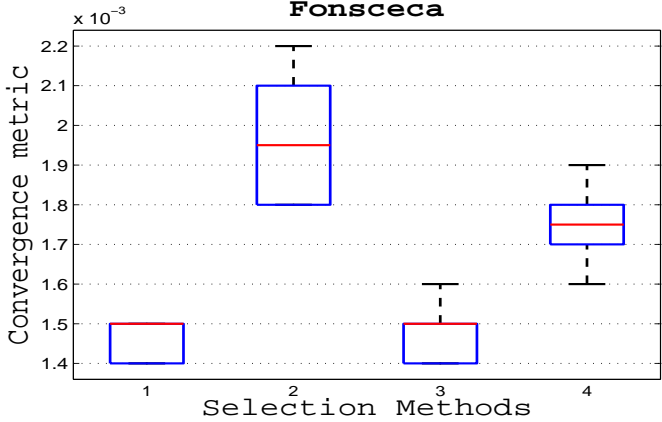
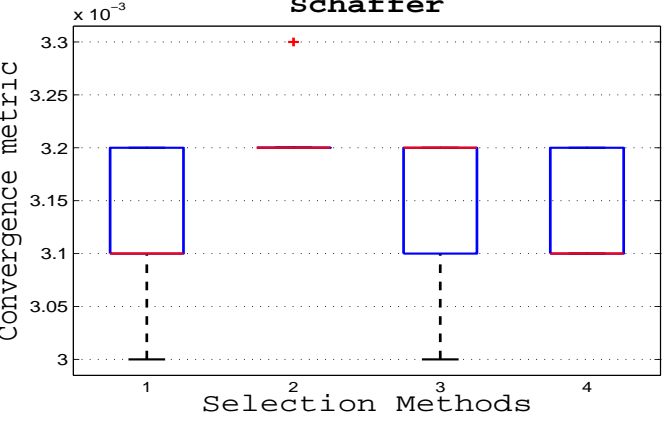
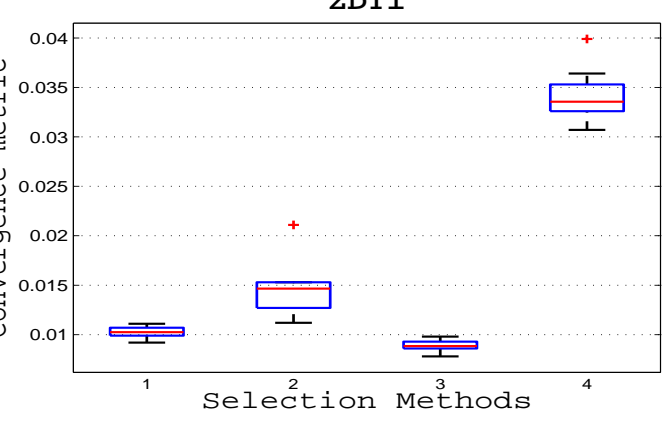
Regarding the diversity metric, as illustrated in Tables 5.5 and 5.6, MOQPSO with the crowding method obtains the best diversity values. However, the proposed hybrid strategy exhibits better results than with the other strategies for ZDT2 problem and intermediate results for the remaining test problems. In other words, It performs better than random and sigma methods in most of the test functions and performs slightly less than the crowding method.



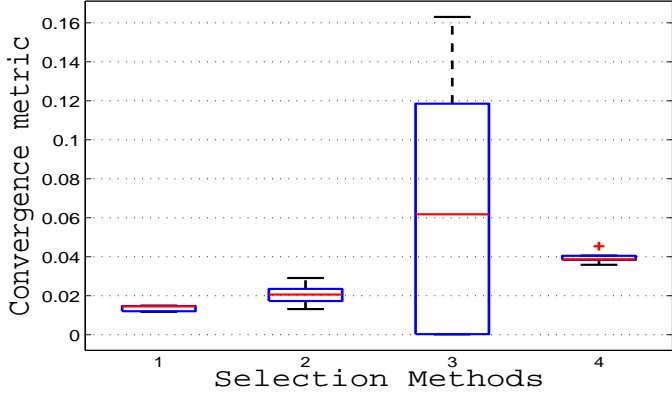
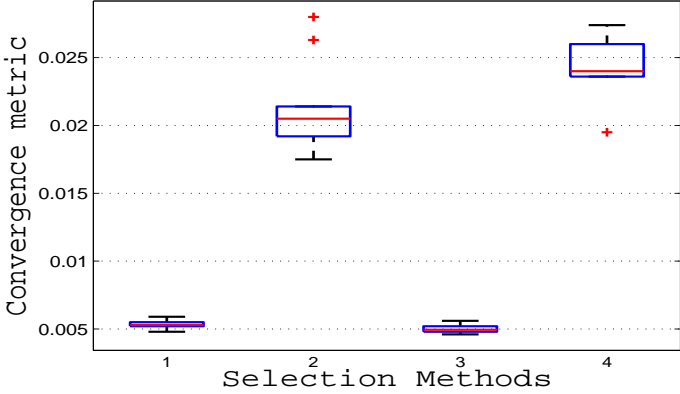
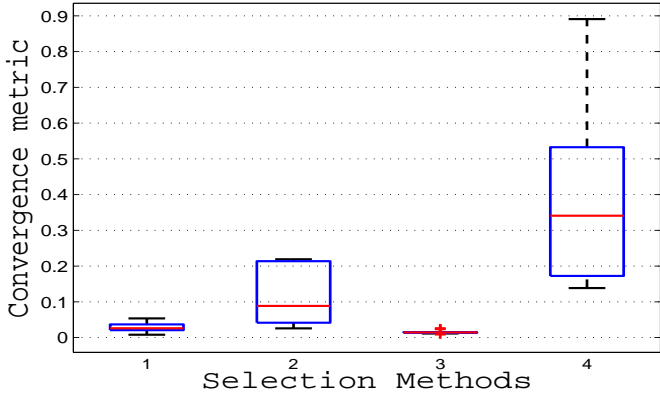
With respect to the p-values in Tables 5.5 and 5.6 and the multicomparison tests in Figure 5.2, the difference between the hybrid method and the other strategies was significant in some cases (on SCH, ZDT3, and ZDT6) compared to the crowding method, on ZDT1 compared to the random method and on ZDT2 compared to the sigma method and not significant in the remaining cases.

In general, MOQPSO with the sigma method recorded good convergence yet poor diversity in most of the test functions compared with hybrid and crowding strategies. On the other hand, MOQPSO with the crowding distance method exhibited good diversity but poor convergence in most of the test functions compared with hybrid and sigma strategies.

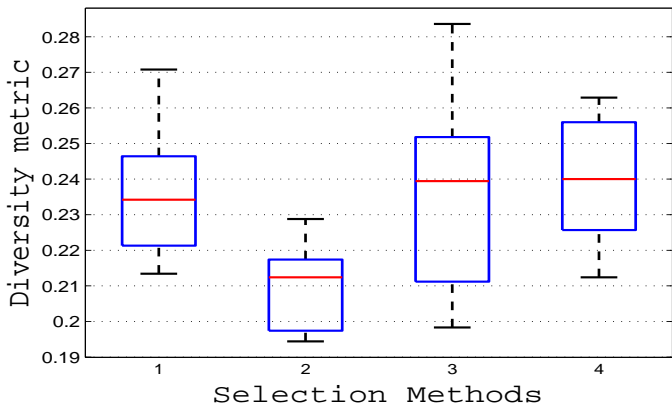
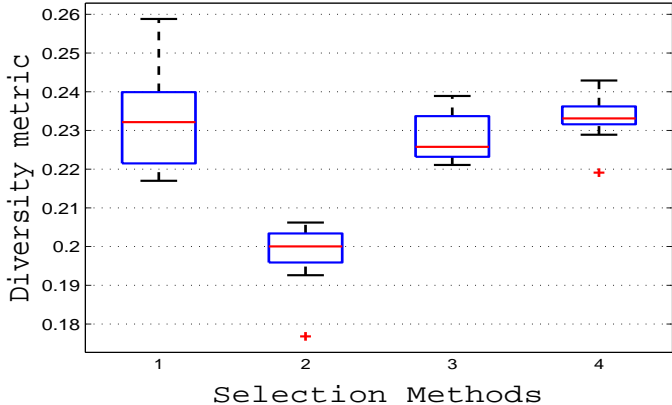
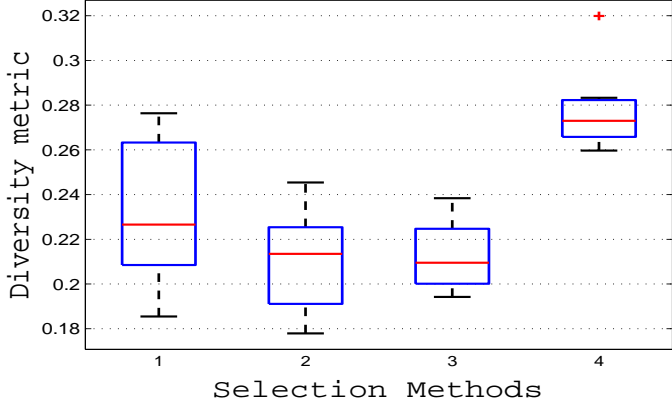
As for the run time, it can be seen from Figure 5.3 that the hybrid method has the highest computational time for FON, ZDT2, and ZDT3 test functions when compared to the other selection methods. On the other hand, the random method shows the least computational time in all the test functions.

Convergence	p-value
 <p><b>Fonsceca</b></p> <p>Convergence metric <math>\times 10^{-3}</math></p> <p>Selection Methods</p>	$5.5355 * 10^{-6}$
 <p><b>Schaffer</b></p> <p>Convergence metric <math>\times 10^{-3}</math></p> <p>Selection Methods</p>	0.0146
 <p><b>ZDT1</b></p> <p>Convergence metric</p> <p>Selection Methods</p>	$1.3801 * 10^{-6}$

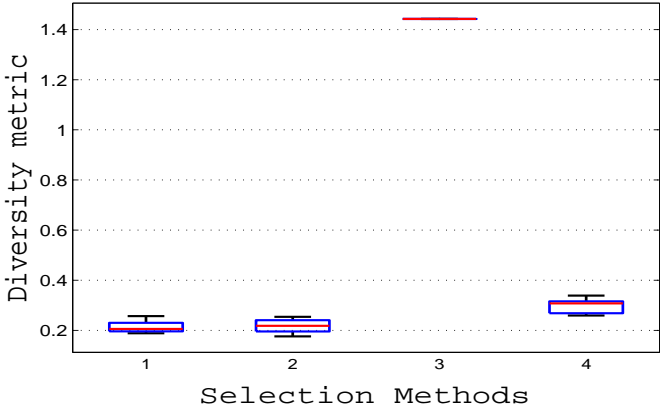
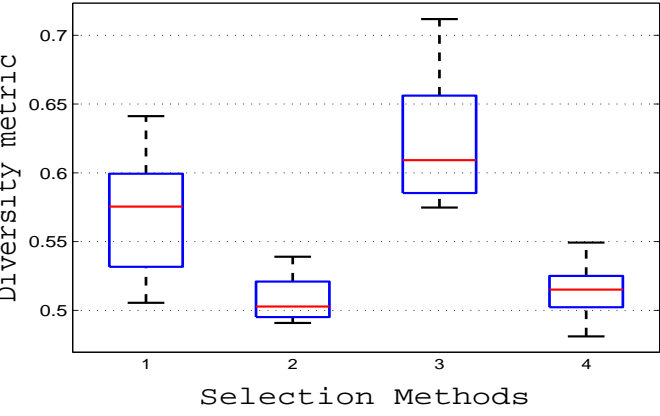
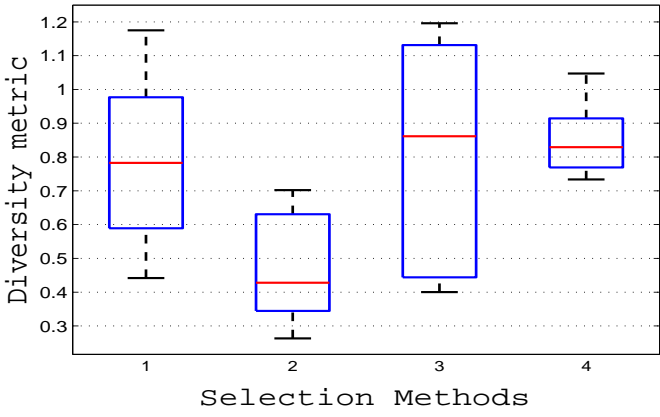
Tabel 5.3: Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to convergence results on unconstrained functions (FON, SCH, ZDT1) with four selection methods based on 1.Hybrid method, 2.Crowding distance, 3.Sigma method, 4.Random selection.

Convergence	p-value
<p><b>ZDT2</b></p>  <p>Convergence metric</p> <p>Selection Methods</p>	0.0019
<p><b>ZDT3</b></p>  <p>Convergence metric</p> <p>Selection Methods</p>	$4.0241 * 10^{-6}$
<p><b>ZDT6</b></p>  <p>Convergence metric</p> <p>Selection Methods</p>	$7.8643 * 10^{-6}$

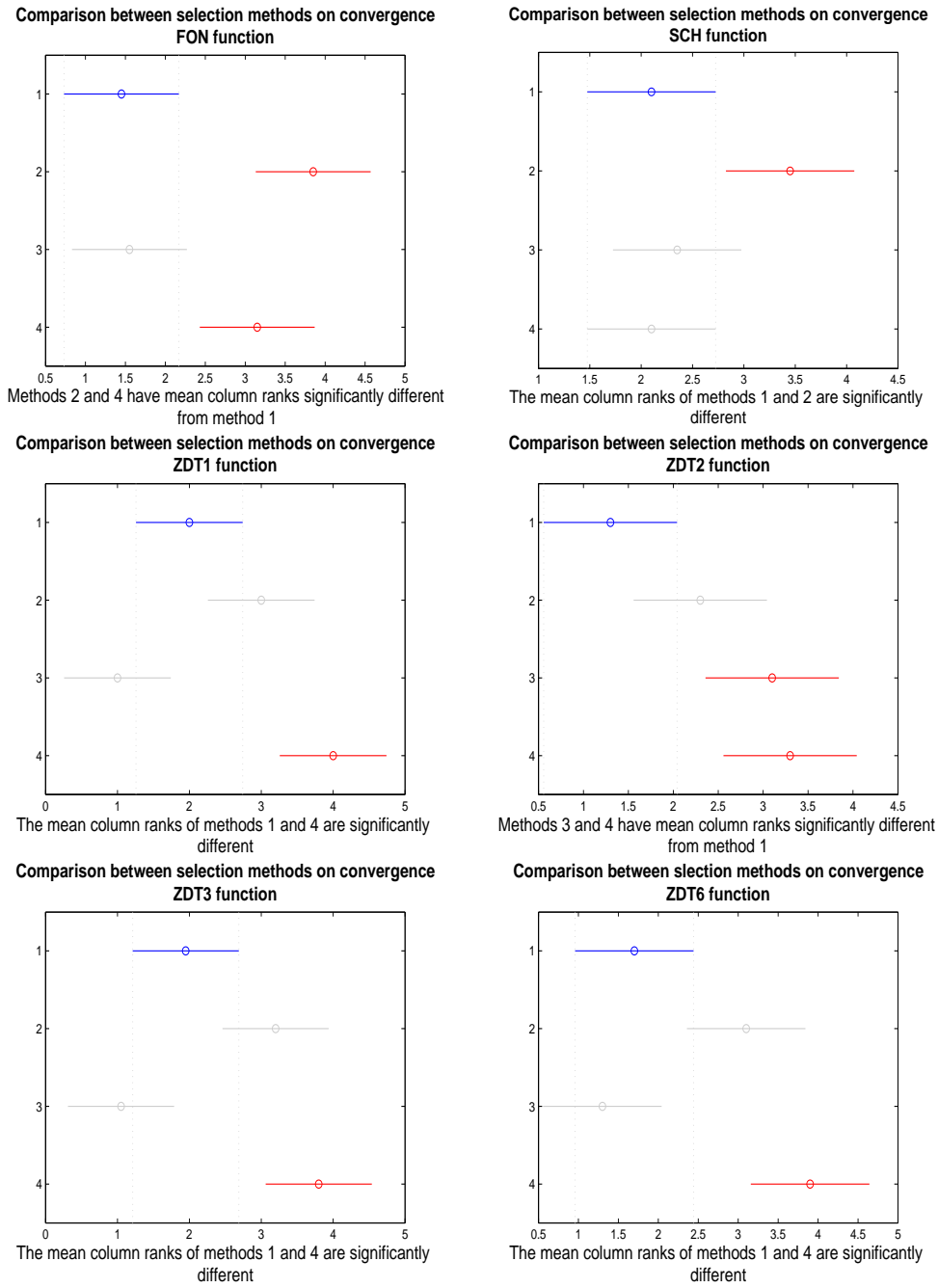
Tabel 5.4: Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to convergence results on unconstrained functions (ZDT2, ZDT3, ZDT6) with four selection methods based on 1.Hybrid method, 2.Crowding distance, 3.Sigma method, 4.Random selection.

Diversity	p-value
<p><b>Fonseca</b></p>  <p>Diversity metric</p> <p>Selection Methods</p>	0.1604
<p><b>Schaffer</b></p>  <p>Diversity metric</p> <p>Selection Methods</p>	0.0003
<p><b>ZDT1</b></p>  <p>Diversity metric</p> <p>Selection Methods</p>	0.0002

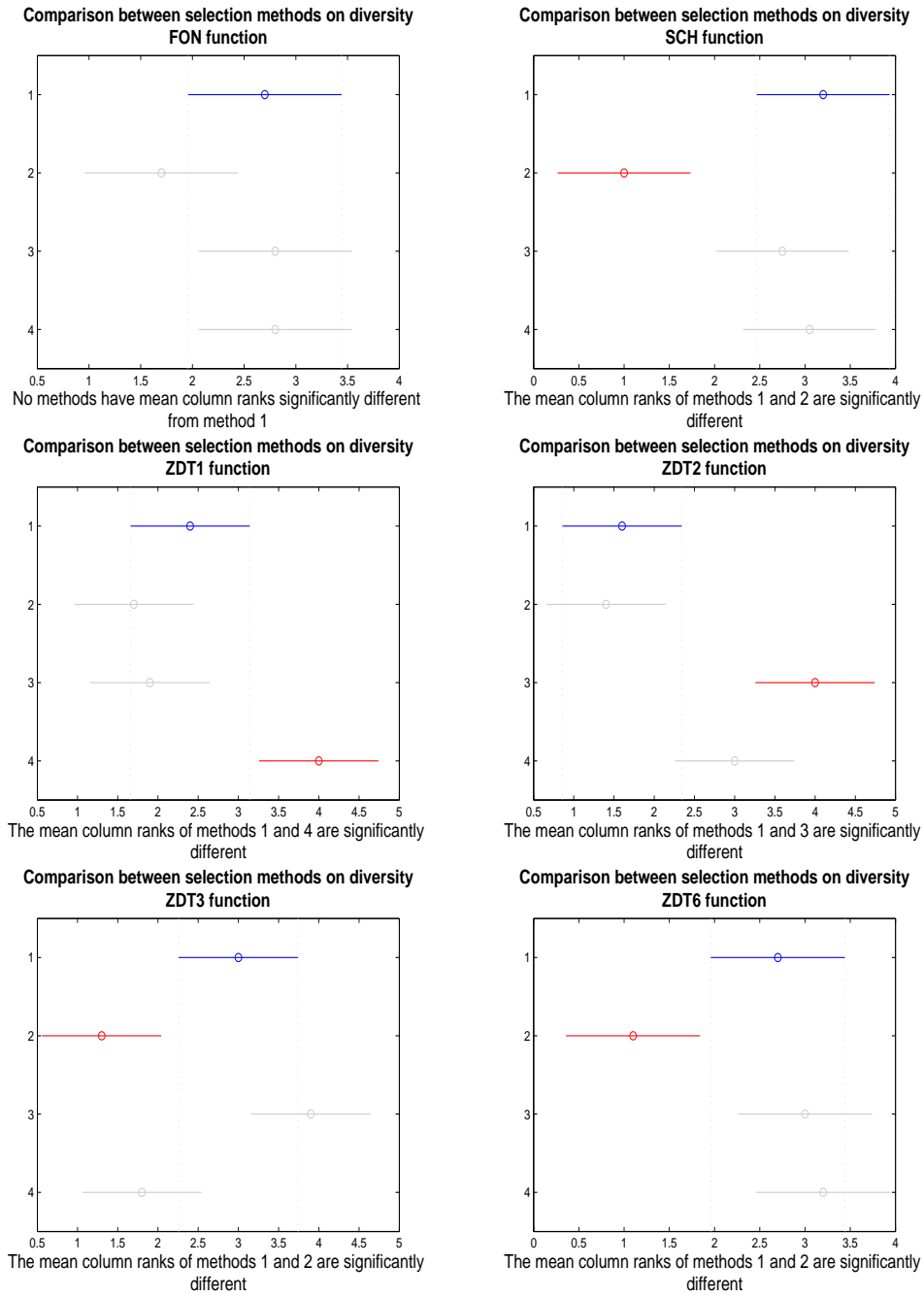
Tabel 5.5: Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to diversity results on unconstrained functions (FON, SCH, ZDT1) with four selection methods based on 1.Hybrid method, 2.Crowding distance, 3.Sigma method, 4.Random selection.

Diversity	p-value
<p><b>ZDT2</b></p>  <p>Diversity metric</p> <p>Selection Methods</p>	$5.5560 * 10^{-6}$
<p><b>ZDT3</b></p>  <p>Diversity metric</p> <p>Selection Methods</p>	$1.6677 * 10^{-5}$
<p><b>ZDT6</b></p>  <p>Diversity metric</p> <p>Selection Methods</p>	0.0009

Tabel 5.6: Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to diversity results on unconstrained functions (ZDT2, ZDT3, ZDT6) with four selection methods based on 1.Hybrid method, 2.Crowding distance, 3.Sigma method, 4.Random selection.



Figur 5.1: Graphs of multicomparison tests of selection methods based on Friedman statistics on convergence values for unconstrained functions. Overlapping intervals indicate no significant difference



Figur 5.2: Graphs of multicomparison tests of selection methods based on Friedman statistics on diversity values for unconstrained functions. Overlapping intervals indicate no significant difference

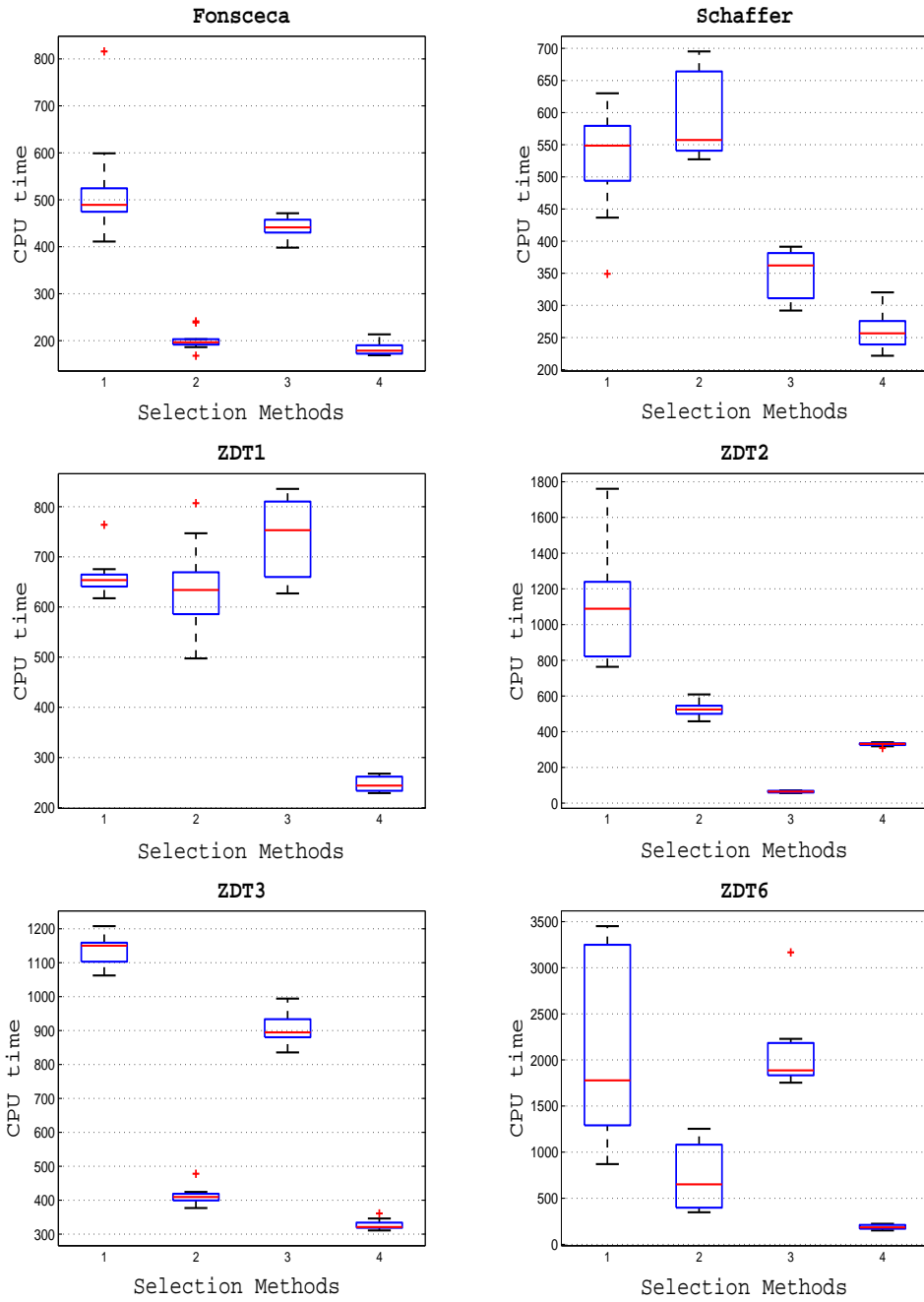


Figure 5.3: Box plots of CPU time for unconstrained functions with four selection methods based on 1. Proposed hybrid method, 2. Crowding distance, 3. Sigma method, 4. Random selection.



For constrained problems, we decided to employ the second constraint handling strategy in this study since the first constraint handling strategy requires more computational time than the second constraint handling strategy and exhibits just a slightly better performance than the second constraint handling strategy. This has been demonstrated in section 4.4.3. Table 5.7 - Table 5.10 illustrate the convergence and the diversity results obtained in the form of box plots together with the p-values when applying the first and the second constraint handling methods with each of the leader selection strategies respectively.

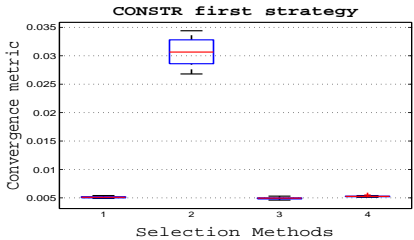
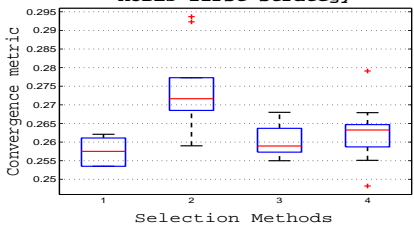
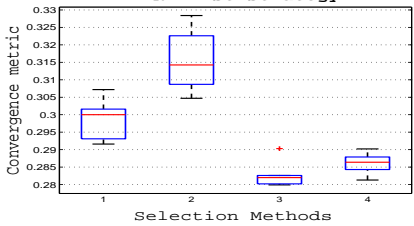
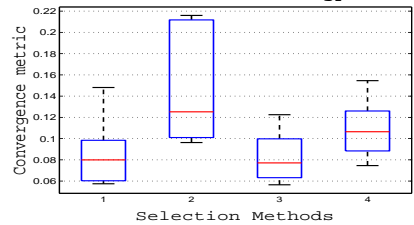
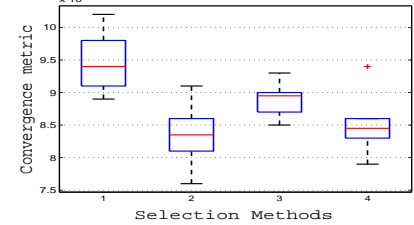
For the first constraint handling strategy, it can be clearly seen from Table 5.7 that CMOQPSO exhibits better convergence results with sigma method and worse convergence results with crowding method. However, the hybrid method outperforms its counterparts in MOBES function and competes with sigma method in achieving best convergence values as in CONSTR and KITA test functions. This fact can be corroborated by the multicomparison test graphs in Figure 5.4 that shows a significant difference between the hybrid and the crowding methods in all cases. In addition, CMOQPSO with the crowding method for TNK test function presents better values in convergence metric than with the other selection methods. This can be attributed to the discontinuous nature of the TNK Pareto front. According to the p-values that have been recorded in Table 5.7 for the convergence metric with the first constraint handling strategy, there is a significant difference between the selection methods for all constrained test functions at the significance level  $\alpha = 0.05$ .

For the diversity metric, as seen in Table 5.8, CMOQPSO shows better performance with the crowding method than with the rest on most of the test functions. The hybrid method is the second best performer in most cases. An exception for TNK function can be observed where CMOQPSO with the hybrid method performs better than with the crowding method. The p-values of the diversity metric presented in Table 5.8 show that there is a

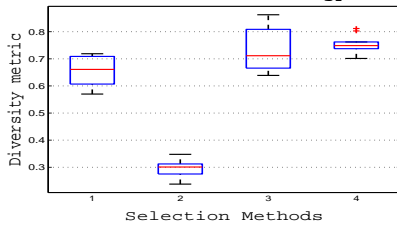
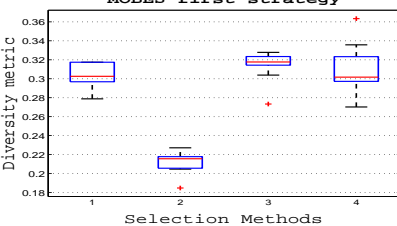
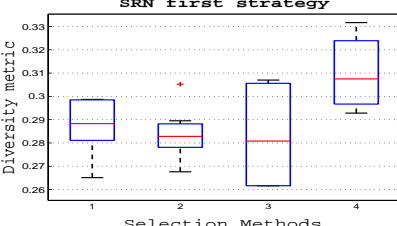
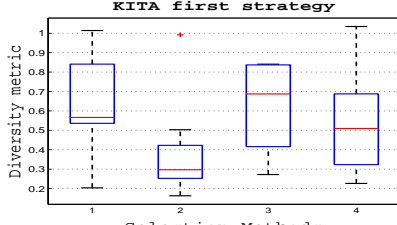
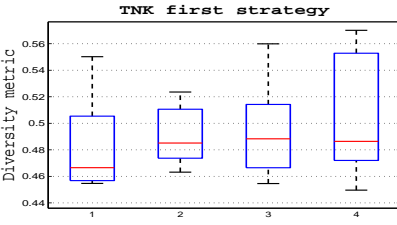
significant difference between the selection methods at the significance level  $\alpha = 0.05$  for all constrained test functions except for TNK function where  $p\text{-value} = 0.3561$ .

With respect to the second constraint handling strategy, we can observe from the box plots presented in Tables 5.9 and 5.10 that CMOQPSO gives best convergence results with the random strategy for CONSTR, SRN, and KITA functions and worst convergence results with the crowding method for CONSTR and SRN. In general, the proposed hybrid selection method competes with the sigma and the random selection strategies in achieving best convergence and competes with the crowding selection method in giving best diversity. An exception can be noticed for MOBES and KITA test functions where CMOQPSO with the crowding method performs slightly better in terms of convergence compared to the hybrid and sigma selection strategies and slightly worse in diversity compared to the sigma method in the case of MOBES function. However, the difference is not significant as shown by the  $p$ -values in Table 5.9 and multicomparison tests in Figure 5.5. Regarding the run time, Figure 5.6 displays the box plots related to the CPU time of the first and the second constraint handling strategies. Generally, it is apparent that the second constraint strategy requires less computational time than the first constraint strategy. This is due to the fact that the first constraint strategy deals only with feasible solutions and neglects any infeasible ones generated during the search process. Therefore, an extra time is spent in generating feasible solutions. For both constraint handling strategies, the hybrid and the sigma methods require more computational time than the crowding and the random selection methods except on CONSTR test function where the random method has a higher computational time than the remaining methods for the second constraint handling strategy.

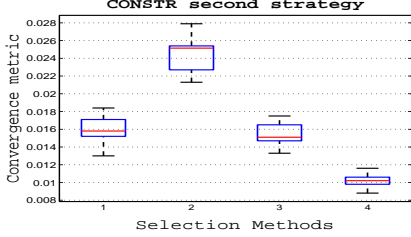
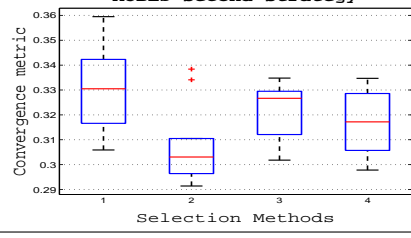
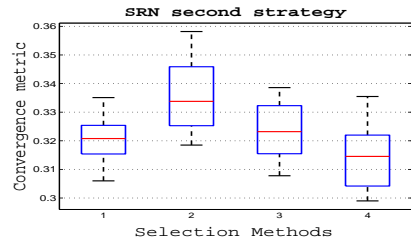
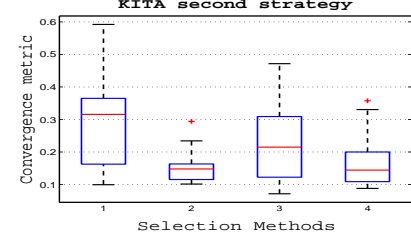
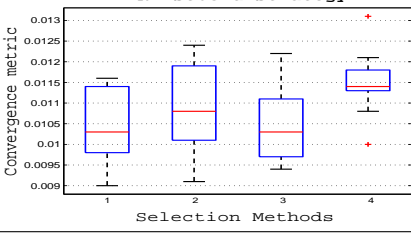
In general, the obtained results for both constrained and unconstrained problems corroborate the fact that the hybrid strategy achieves a balance between convergence and diversity compared to the sigma based strategy and the crowding distance based strategy.

Convergence	p-value
 <p><b>CONSTR first strategy</b></p> <p>Convergence metric</p> <p>Selection Methods</p>	$3.5427 * 10^{-6}$
 <p><b>MOBES first strategy</b></p> <p>Convergence metric</p> <p>Selection Methods</p>	0.0027
 <p><b>SRN first strategy</b></p> <p>Convergence metric</p> <p>Selection Methods</p>	$3.4943 * 10^{-6}$
 <p><b>KITA first strategy</b></p> <p>Convergence metric</p> <p>Selection Methods</p>	0.0087
 <p><b>TNK first strategy</b></p> <p>Convergence metric</p> <p>Selection Methods</p>	$2.6907 * 10^{-5}$

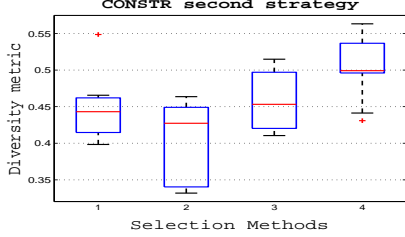
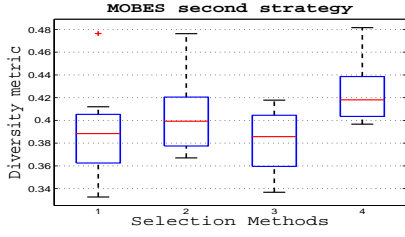
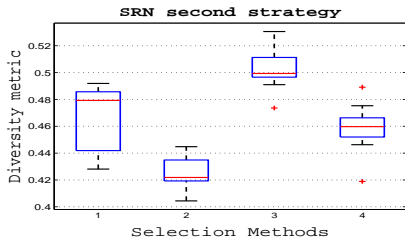
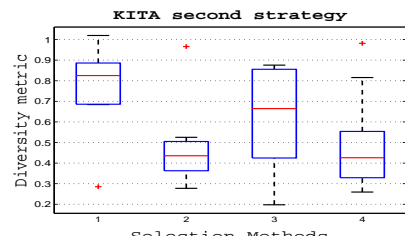
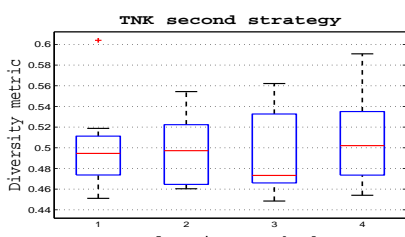
Tabel 5.7: Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to convergence results for constrained functions with first constraint strategy on four selection methods based on 1.Hybrid method, 2.Crowding distance, 3.Sigma method, 4.Random selection.

Diversity	p-value
	0.0001
	$9.3859 * 10^{-5}$
	0.0074
	0.0160
	0.3561

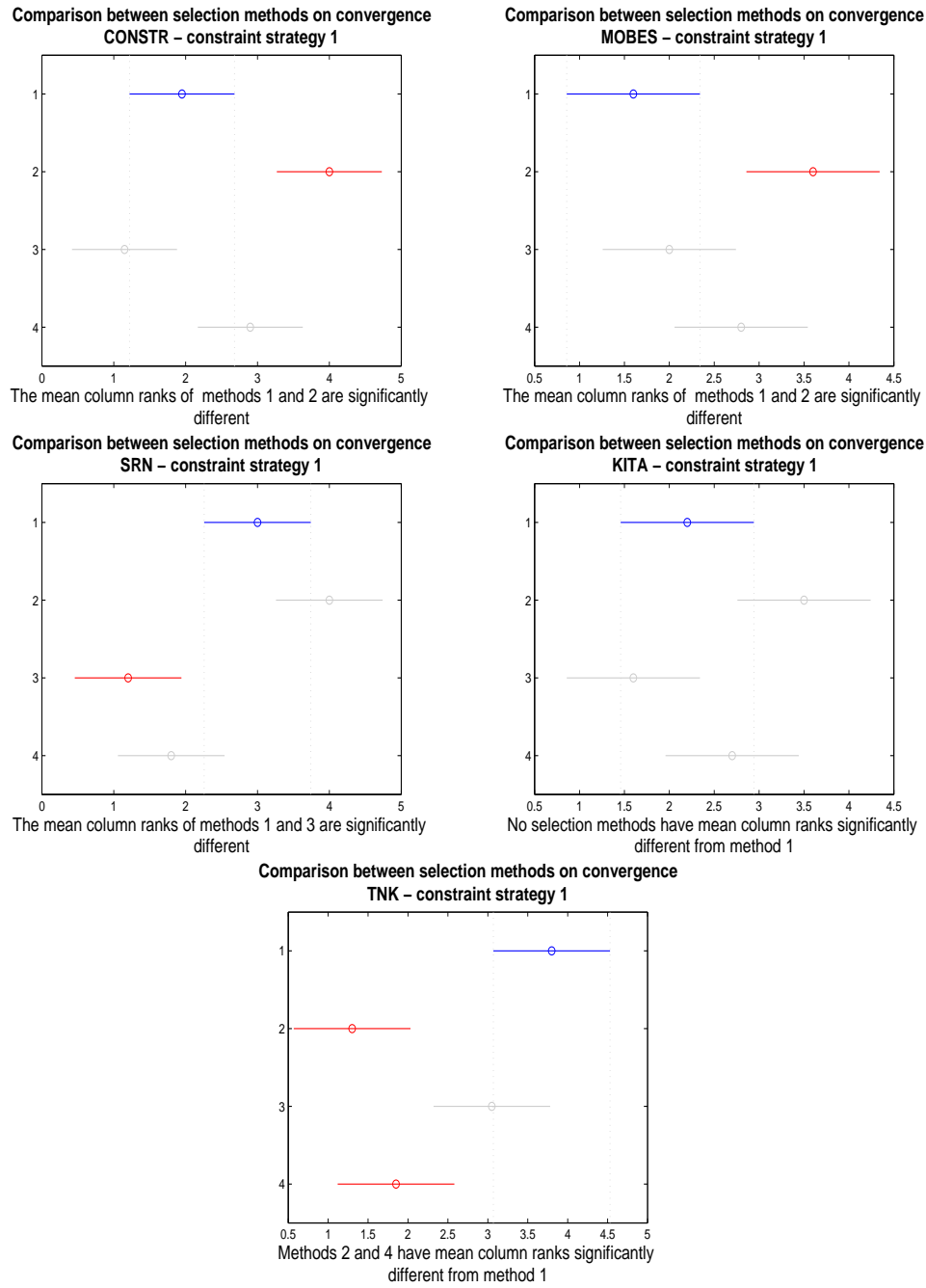
Tabel 5.8: Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to diversity results for constrained functions with first constraint strategy on four selection methods based on 1.Hybrid method, 2.Crowding distance, 3.Sigma method, 4.Random selection.

Convergence	p-value
 <p>CONSTR second strategy</p>	$4.6694 * 10^{-6}$
 <p>MOBES second strategy</p>	0.1176
 <p>SRN second strategy</p>	0.0109
 <p>KITA second strategy</p>	0.0732
 <p>TNK second strategy</p>	0.2096

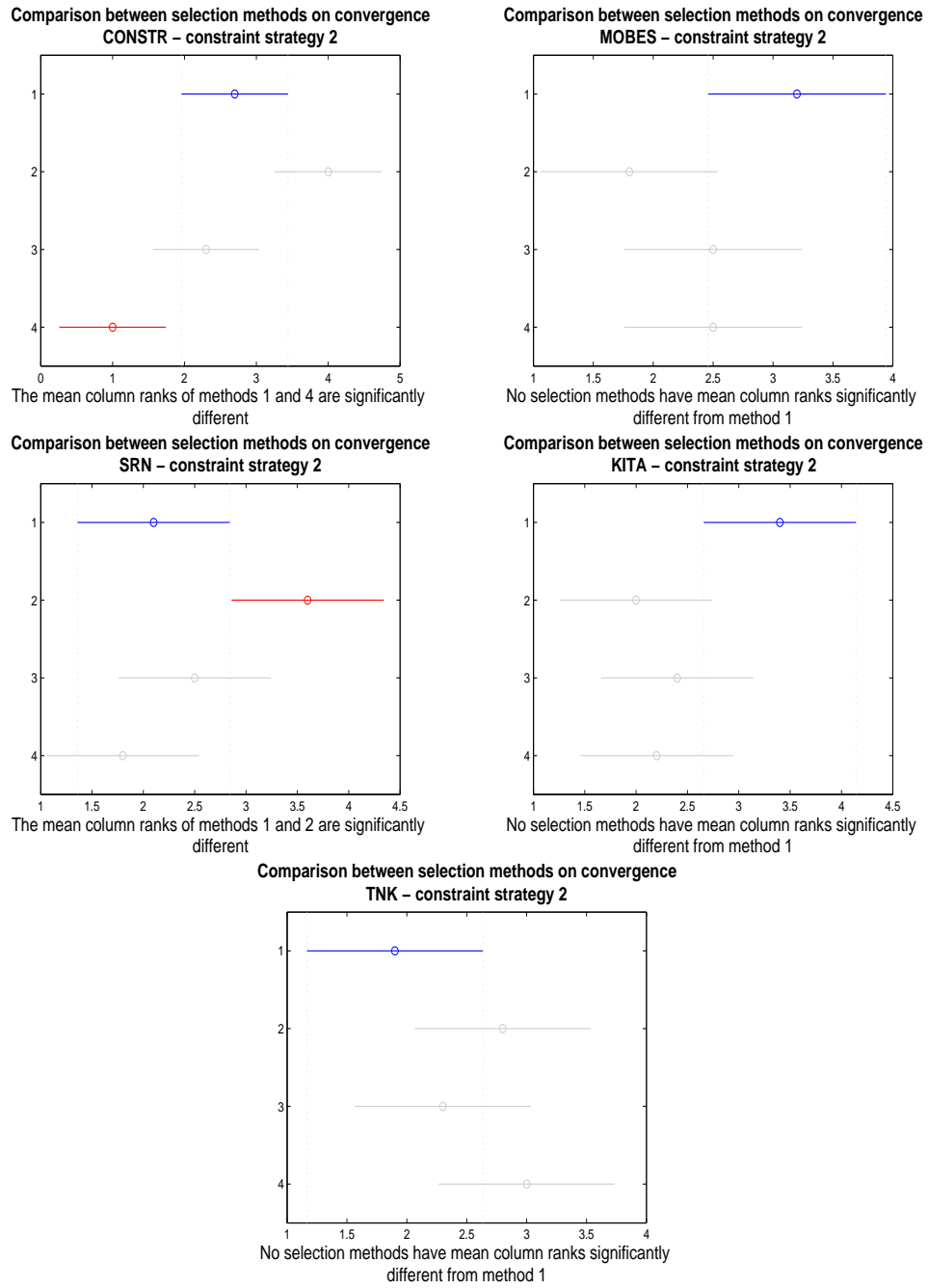
Tabel 5.9: Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to convergence results for constrained functions with second constraint strategy on four selection methods based on 1.Hybrid method, 2.Crowding distance, 3.Sigma method, 4.Random selection.

Diversity	p-value
	0.0211
	0.0658
	$4.7085 * 10^{-5}$
	0.0327
	0.7530

Tabel 5.10: Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to diversity results for constrained functions with second constraint strategy on four selection methods based on 1.Hybrid method, 2.Crowding distance, 3.Sigma method, 4.Random selection.

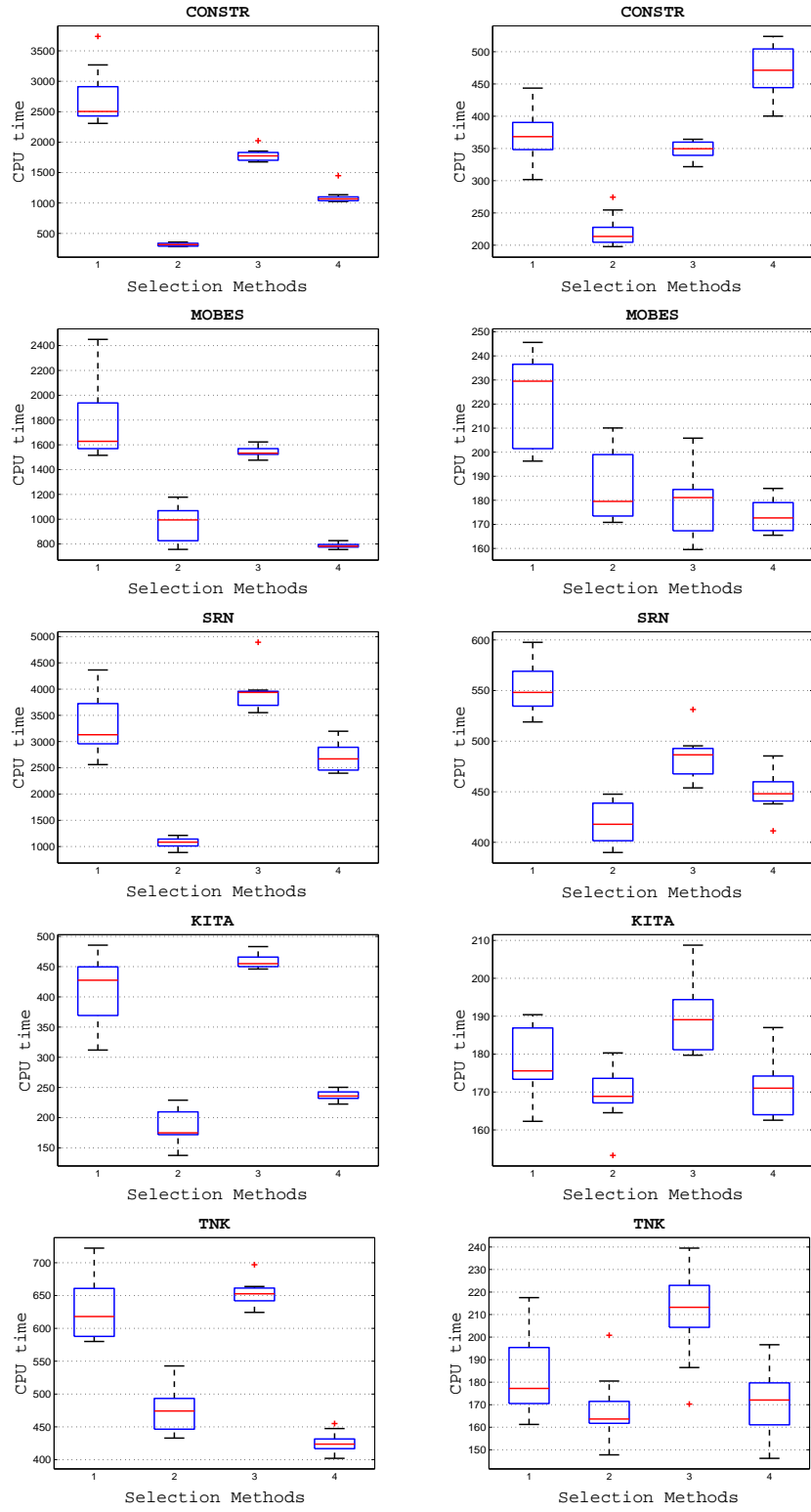


Figur 5.4: Graphs of multicomparison tests of selection methods based on Friedman statistics on convergence values for the first constraint strategy. Overlapping intervals indicate no significant difference.



Figur 5.5: Graphs of multicomparison tests of selection methods based on Friedman statistics on convergence values for the second constraint strategy. Overlapping intervals indicate no significant difference.





Figur 5.6: Box plots of CPU time on constrained functions with four selection methods based on 1.Hybrid method, 2.Crowding distance, 3.Sigma method, 4.Random selection. First constraint strategy (left column) and second constraint strategy (right column).

### 5.1.2 Experiment2: MOQPSO vs MOPSO

In order to compare QPSO with PSO in multi-objective optimization context the following procedure has been employed. MOPSO has been implemented with the same leader selection strategy (the proposed hybrid strategy) and the same archiving methods (redundancy removal for unconstrained problems and unbounded archive for constrained problems). The archiving methods will be explained in chapter 6. We used the proposed second constraint handling strategy, described in section 4.2.2, to handle the constrained test problems in this experiment. Note that like QPSO, a basic version of PSO has been considered that is no mutation operator has been introduced. In Tables 5.13 and 5.14, we summarize the results of the convergence and the diversity metrics for constrained and unconstrained problems obtained over 10 runs for each algorithm. Furthermore, to study the significance of differences between the two algorithms, Wilcoxon rank sum test equivalent to the Mann-Whitney U test has been performed [34]. The meaning of these metrics is explained in section 4.4.3. For PSO parameters, we adopted the most commonly used settings in the literature [22]. In all cases, the PSO parameters have been set as follows:  $W=0.4$ ,  $C1=1.5$ , and  $C2=1.5$ . The number of iterations and the number of particles used in this experiment are presented in Table 5.11 for unconstrained test functions and Table 5.11 for constrained test problems with  $P_G=0.5$  and  $P_S=0.3$ . These settings were obtained based on the same reasoning explained in sections 3.6.3 and 4.4.1.

From the results displayed in Tables 5.13 and 5.14 we can clearly observe the high competitiveness of QPSO with PSO. For unconstrained problems, MOPSO performs slightly better than MOQPSO in terms of convergence. However, MOPSO performs very poorly in the case of ZDT2 function due to the premature convergence problem which makes the algorithm converge to a single solution. That is why no results have been reported in Table 5.14 for ZDT2. While MOQPSO is less likely to get stuck in a local optimum

when solving ZDT2 and hence it achieves better results. In addition, MOQPSO performs slightly better than MOPSO for SCH function.

From diversity point of view, MOQPSO outperforms MOPSO in all ZDT functions and achieves competitive results in FON and SCH functions.

For constrained functions, we can notice a slight advance of MOPSO over MOQPSO except for SRN and CONSTR where MOQPSO shows better convergence results in terms of average and median.

In all cases, MOQPSO achieves results that are significantly different from those of MOPSO except in the case of diversity of FON and MOBES fronts and convergence of CONSTR front.

Test function	Population size	Number of Iterations	$\beta$	k
FON	300	60	1.2 - 0.5	10
SCH	150	50	1.2 - 0.5	10
ZDT1	100	250	1.2 - 0.5	10
ZDT2	100	250	1.2 - 0.5	10
ZDT3	100	250	1.2 - 0.5	10

Tabel 5.11: Parameter settings of MOPSO and MOQPSO for unconstrained test problems.

Test function	Population size	Number of Iterations	$\beta$	k
CONSTR	150	500	1.2 - 0.5	10
MOBES	150	500	1.2 - 0.5	10
SRN	150	500	1.2 - 0.5	10

Tabel 5.12: Parameter settings of MOPSO and MOQPSO with second constraint handling strategy for the constraint test functions.

Test Problem	Statistics	Convergence			Diversity		
		MOPSO	MOQPSO	P-value	MOPSO	MOQPSO	P-value
SRN	Best	0.3063	0.3060	$1.823 * 10^{-4}$	0.3360	0.3407	$1.816 * 10^{-4}$
	Worst	0.3326	0.3351		0.3799	0.3981	
	Average	0.3217	0.3198		0.3587	0.3789	
	Median	0.3236	<b>0.3208</b>		<b>0.3628</b>	0.3828	
	Std.	0.0086	0.0085		0.0139	0.0177	
CONSTR	Best	0.0123	0.0130	0.545	0.3091	0.3984	$1.827 * 10^{-4}$
	Worst	0.0196	0.0184		0.3764	0.5486	
	Average	0.0164	0.0159		0.3383	0.4444	
	Median	0.0173	0.0158		<b>0.3377</b>	0.4431	
	Std.	0.0025	0.0016		0.0216	0.0439	
MOBES	Best	0.2837	0.3059	$1.776 * 10^{-4}$	0.3604	0.3326	0.6770
	Worst	0.3198	0.3449		0.4530	0.4764	
	Average	0.3021	0.3281		0.3988	0.3899	
	Median	<b>0.3013</b>	0.3305		0.3874	0.3884	
	Std.	0.0126	0.0138		0.0355	0.0389	

Tabel 5.13: Results of MOPSO and MOQPSO for constrained test functions in terms of statistics on convergence, diversity and the p-value with respect to ( $p < \alpha = 0.05$ ).

Test Problem	Statistics	Convergence			Diversity		
		MOPSO	MOQPSO	P-value	MOPSO	MOQPSO	P-value
FON	Best	0.00010	0.0019	$1.688 * 10^{-4}$	0.2159	0.1901	0.3057
	Worst	0.00089	0.0024		0.2575	0.2846	
	Average	0.00093	0.0021		0.2398	0.2448	
	Median	<b>0.00095</b>	0.0021		0.2430	0.2531	
	Std.	0.00004	0.0002		0.0119	0.0325	
SCH	Best	0.0033	0.0030	$1.086 * 10^{-4}$	0.2056	0.2238	0.0072
	Worst	0.0035	0.0032		0.2398	0.2561	
	Average	0.0034	0.0031		0.2223	0.2382	
	Median	0.0034	<b>0.0031</b>		<b>0.2218</b>	0.2313	
	Std.	0.0001	0.0001		0.0096	0.0125	
ZDT1	Best	0.0034	0.0108	$1.786 * 10^{-4}$	0.4865	0.2315	$1.786 * 10^{-4}$
	Worst	0.0076	0.0135		0.9183	0.2782	
	Average	0.0057	0.0122		0.7478	0.2571	
	Median	<b>0.0060</b>	0.0123		0.7425	<b>0.2574</b>	
	Std.	0.0019	0.0010		0.1249	0.0151	
ZDT2	Best	-	0.0111	$6.340 * 10^{-4}$	-	0.1897	$6.386 * 10^{-5}$
	Worst	-	0.0136		-	0.2644	
	Average	-	0.0126		-	0.2313	
	Median	-	<b>0.0129</b>		-	<b>0.2351</b>	
	Std.	-	0.0009		-	0.0221	
ZDT3	Best	0.0034	0.0046	$1.278 * 10^{-4}$	0.8006	0.5244	$1.285 * 10^{-4}$
	Worst	0.0036	0.0058		0.8186	0.6694	
	Average	0.0035	0.0051		0.8132	0.5967	
	Median	<b>0.0034</b>	0.0051		0.8186	<b>0.5863</b>	
	Std.	0.0001	0.0004		0.0087	0.0527	

Tabel 5.14: Results of MOPSO and MOQPSO for unconstrained test functions in terms of statistics on convergence, diversity and the p-value with respect to ( $p < \alpha = 0.05$ ).

## 5.2 Summary

From the experiments discussed in this chapter, we can draw the following conclusions:

- For unconstrained test problems:
  - The MOQPSO with the sigma method exhibits best convergence results yet low diversity in most of the test functions. On the other hand, MOQPSO with the crowding method records best diversity values yet poor convergence in most of the test functions.
  - The proposed hybrid method competes with the sigma method in obtaining best convergence results and competes with the crowding method in achieving best diversity values. In other words, the hybrid method shows its ability in maintaining a balance in obtaining good convergence and good diversity values. The hybrid method even exhibits better convergence results than the sigma method in some of the test functions specially in the case of ZDT2 function where the sigma method failed to solve this function due to the premature convergence problem.
  - Regarding the computational time, the proposed hybrid method has the highest computational time in some of the test functions. On the other hand, the random method shows the least computational time in all of the test functions.
- For constrained test problems:
  - The first and the second constraint handling strategies show similar results in that CMOQPSO with the sigma method exhibits better convergence results and CMOQPSO with the crowding method shows better diversity results.

- As expected, the proposed hybrid method competes with the sigma method in achieving best convergence and competes with the crowding method in obtaining best diversity values. It outperforms the crowding method in terms of convergence and outperforms the sigma method in terms of diversity.
  - The first strategy outperforms the second strategy in obtaining best convergence and diversity results in all of the test functions.
  - The first strategy requires more computational time than the second constraint strategy.
- For comparison of PSO against QPSO:
    - No variant has shown to be the best in all unconstrained and constrained test problems from both convergence and diversity points of views.
    - MOPSO failed to solve ZDT2 test function due to the premature convergence problem while MOQPSO obtained the best performance when solving the same function.

It can be clearly observed that the rationale behind the proposed hybrid selection strategy in achieving good convergence using the sigma method while maintaining good diversity using the crowding distance method has been experimentally confirmed. Our experiments have shown that the best convergence results have been recorded with the sigma selection method and the best diversity results have been obtained with the crowding selection method. However, the sigma method fails to obtain good diversity and the crowding method fails to maintain good convergence.

### **An Extensive Empirical Study of the Impact of Different Archiving Strategies for Constrained and Unconstrained MOPs**

---

In this chapter,<sup>1</sup> we will introduce a new unbounded archive strategy for handling the archive size of the non-dominated solutions, which we called redundancy removal. This strategy will be described and then compared with other archiving methods, namely the unbounded archive method, the clustering method, the crowding method, and the maximin method. An extensive experimental study is performed to study the influence of several archiving methods on the performance of proposed MOQPSO with respect to convergence, diversity, and computational time. The experiments are designed to answer the following research questions:

1. What is the impact of the archiving strategies on the algorithm performance in terms of convergence, diversity, and CPU time?

---

<sup>1</sup>A shorter version of the work in this chapter is included in the following submission: Heyam Al-Baity, Souham Meshoul, and Ata Kaban. Swarm Based Multi-Objective Optimization with Quantum Behaved Particles. International Journal of Bio-Inspired Computation (IJBIC), submitted, 2014.



2. What is the impact of the constraint handling strategies on the performance of MOQPSO with the different archiving methods?

## 6.1 Handling Archive Size

The archive size of the non-dominated solutions is a critical parameter as it influences the performance of the algorithm and the quality of the obtained fronts. If  $L^t = |GBA^t|$  is the size of the archive at iteration  $t$ , the number of tests for domination in the Update-Archive procedure, described in section 3.4, in the worst case scenario is equal to  $(N * L^t)$  where  $N$  is the number of particles. The archive can be of bounded or unbounded size. The unbounded archive may lead to good quality Pareto fronts at the expense of a growing run time as the size would grow large through iterations. On the other hand, with a bounded archive, the number of the stored non-dominated solutions will be reduced to the bounding size. In this case, an archiving strategy is required to filter the archive in a way to maintain its size fixed whenever the archive becomes full. Therefore, the matter is how to determine a suitable size of the archive to obtain a good balance between run time and quality of the obtained fronts. Hence, with a bounded size, the algorithm may be less time demanding provided that the used archiving strategy itself is not computationally expensive. However, limiting the size of the archive may impact the quality of the obtained solutions [77][79]. There are several bounded archiving methods in the literature. A review can be found in [61][79].

To study the impact of the archiving methods on our proposed MOQPSO, we considered three popular state-of-the-art archiving methods, namely clustering [118], crowding [32] and maximin [69]. The details of clustering and maximin archiving methods are given by the two procedures described shortly in Algorithms 16 and 17. For the crowding method, the principle is to sort solutions according to their crowding distance values

computed as described in section 3.3 and keep those solutions that are less crowded.

### 6.1.1 The Proposed Redundancy Removal Archiving Strategy

In this section, we propose a new simple yet effective method to handle the archive size of the non-dominated solutions, which we call redundancy removal archiving strategy. Its aim is to reap advantage from both policies, i.e., bounded and unbounded size. That is, getting the advantage of the unbounded archive in reaching good convergence to the Pareto front and maintaining diversity while reducing the computational time of the algorithm and providing less number of final non-dominated solutions compared to the unbounded policy. The underlying idea is to use an unbounded archive size while keeping the archive redundancy free. That is why we call this archiving strategy as redundancy removal method. Periodically during the search process, the archive undergoes a filtering operation that removes redundant solutions. A solution is said to be redundant depending on its closeness to the other solutions in the archive based on a threshold value. The metric used to compute the closeness between solutions is the usual Euclidean distance. The outline of the redundancy removal archiving method is described in Algorithm 15.

---

#### Algorithm 15 Redundancy-Removal (A)

---

```

1: Input: Current set of non-dominated solutions A
2:  $NewArchive(1) = A(1)$ 
3: for  $i = 2$  to  $|A|$  do
4:    $Found = FALSE$ 
5:   for  $j = 1$  to  $|NewArchive|$  do
6:     if  $Norm(A(i) - NewArchive(j)) \prec Threshold$  then
7:        $FOUND = TRUE$ 
8:       Break { exit For j loop if redundant solution is found }
9:     end if
10:  end for
11:  if  $NotFound$  then
12:     $NewArchive(|NewArchive| + 1) = A(i)$  {insert A(i) solution into NewArchive}
13:  end if
14: end for
15: Output : NewArchive

```

---

---

**Algorithm 16** Clustering (A)[79]

---

```
1: Input: Current set of non-dominated solutions A
2:  $Z = |A|$ ;
3: while ( $Z > MaxSize$ ) do
4:    $x1, x2 = FindClosestPair(A)$ 
5:    $d1 = FindSecondClosest(A, x1)$ 
6:    $d2 = FindSecondClosest(A, x2)$ 
7:   if  $d1 < d2$  then
8:     remove x1 from A
9:   else
10:    remove x2 from A
11:   end if
12:    $Z = Z - 1$ 
13: end while
```

---

---

**Algorithm 17** Maximin(A)[79]

---

```
1: Input: Current set of non-dominated solutions A
2:  $Z = |A|$ 
3: for  $i = 1$  to  $Z$  do
4:   for  $j = 1$  to  $Z$  do
5:     if  $j \neq i$  then
6:       for  $k = 1$  to  $M$  do
7:          $Difference(k) = F_i(k) - F_j(k)$  {  $M$  is the number of objectives }
8:       end for
9:        $MinValues(j) = \min(Difference)$ 
10:    end if
11:     $A(i).MaximinValue = \max(MinValues)$ 
12:  end for
13: end for
14: Sort (A, MaximinValue, Ascending)
15:  $A = A(1 : MaxSize)$  {MaxSize is the archive size}
```

---

## 6.2 Time Complexity Analysis of the Archiving Methods

In this section, we will describe the time complexity of the archiving methods used in this study.

### Time Complexity of Redundancy Removal Archiving Method

The time complexity of the proposed redundancy removal algorithm can be measured in terms of the number of comparisons used to detect redundancy. According to Algorithm 15, this number is as follows:

$$(1 + 2 + 3 + \dots + (L - 1)) * M = ML(L - 1)/2$$

where  $L$  is the archive size and  $M$  is the number of objective functions. Therefore the overall time complexity of redundancy removal is  $O(ML^2)$ . It should be noted that redundancy removal occurs at periods of time and not at each iteration of the algorithm. That is, it is performed  $(Maxiter/period)$  times where  $Maxiter$  is the maximum number of iterations and  $period$  is the number of iterations up to which redundancy removal is triggered.

### Time Complexity of Clustering Archiving Method

The clustering archiving algorithm has been described in Algorithm 16. According to this algorithm,  $MaxSize$  operations are performed to find the closest pairs where  $MaxSize$  is the archive size. At the beginning, we need to identify the smallest distance among  $L(L-1)/2$ , then among  $(L-1)(L-2)/2$ , and so on until among  $MaxSize(MaxSize+1)/2$ .

Therefore, the overall time complexity of this algorithm is:

$$O((\frac{1}{2}L(L-1) + \frac{1}{2}(L-1)(L-2) + \dots + \frac{1}{2}MaxSize(MaxSize+1)) * M)$$

which can be approximated by  $O(ML^2)$ .

## Time Complexity of Maximin Archiving Method

The maximin archiving algorithm described in Algorithm 17 shows that the main operations consist of calculating the maximin values of each archive member and then sorting them according to these values. The time complexity of computing the maximin values is  $O(ML^2)$  and the time complexity of the sorting process is  $O(L\log L)$ . Therefore, the overall time complexity of the maximin algorithm is  $O(ML^2)$ .

## Time Complexity of Crowding Archiving Method

The crowding based archiving procedure consists of computing the crowding distance value for each archive member the complexity of which is  $O(ML\log L)$  then sorting them in descending order according to the crowding distance values the complexity of which is  $O(L\log L)$ . Therefore, the time complexity of the crowding based archiving method is  $O(ML\log L + L\log L)$ , which leads to  $O(ML\log L)$  [32].

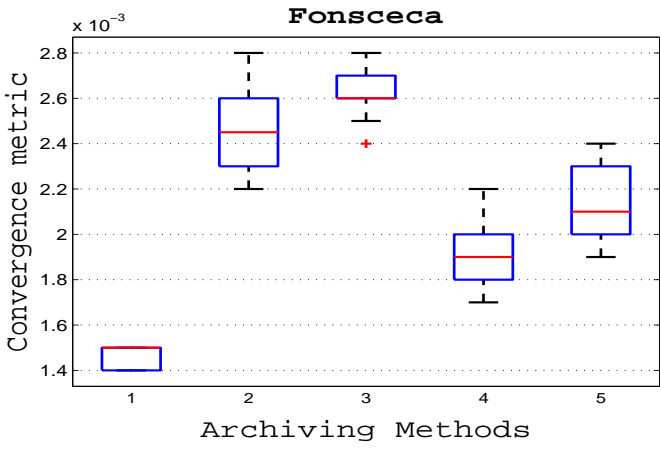
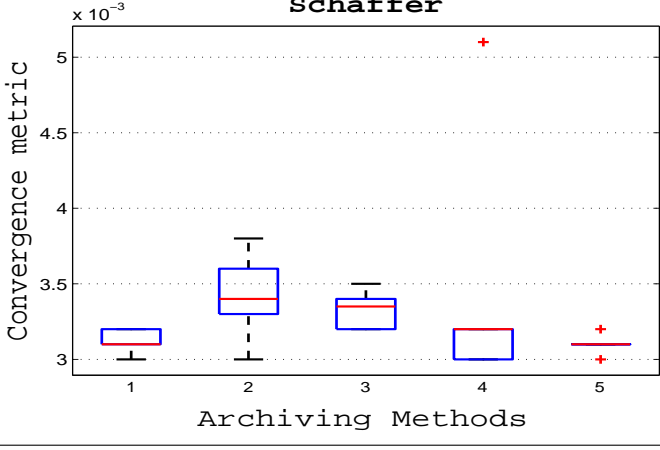
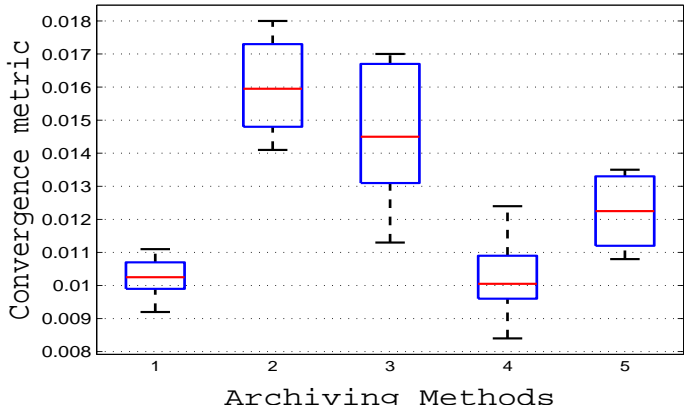
As can be seen, all archiving strategies scale linearly with the number of objective functions  $M$ . However, the crowding based archiving strategy scales quasi-linearly with the archive size while the other strategies have a quadratic scaling with the archive size. Therefore, better time complexity is achieved with the crowding based archiving strategy.

## 6.3 Experiment: Impact of archiving methods

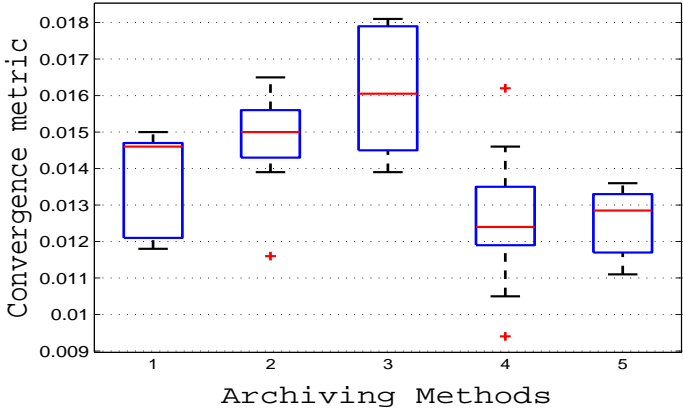
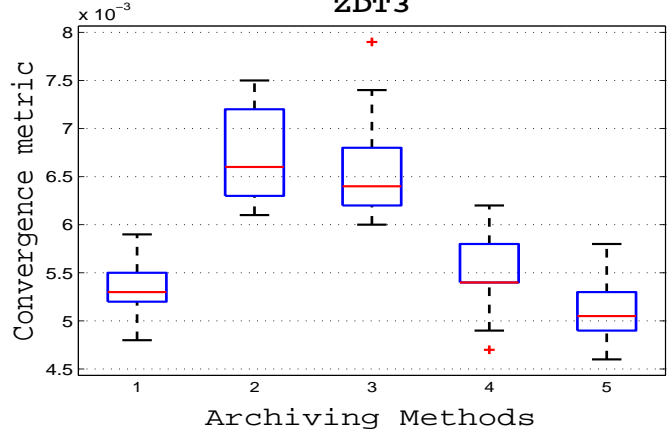
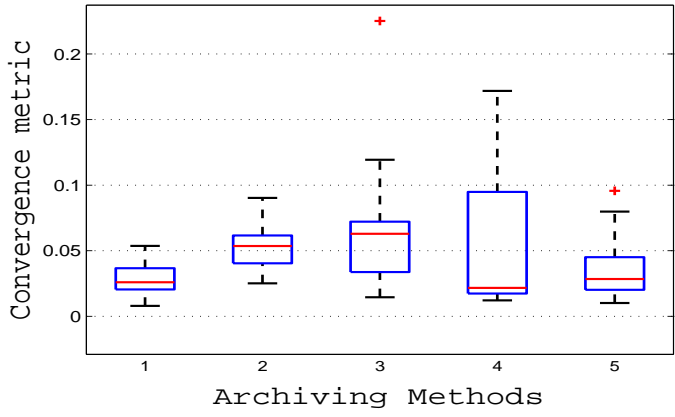
In order to study the behavior of MOQPSO under varying archiving strategies, this experiment has been conducted with the aim to compare its performance with five archiving strategies:

1. The unbounded archive size strategy
2. The clustering archive strategy
3. The crowding archive strategy
4. The maximin archive strategy
5. The proposed redundancy removal archive strategy

We used in this study the same test functions as in the previous chapters for the same reasons explained in section 3.6.1 for unconstrained problems and section 4.4 for constrained problems. All experiments are conducted following the same parameter configurations of the experiments of the leader selection strategies presented in section 5.1.1 with archive size limit =100. The hybrid selection method is the method employed in these experiments for selecting the *gbest* particle. For each scenario, we keep the archiving strategy as the only difference between the five scenarios. Convergence, diversity, and CPU time have been recorded for both unconstrained and constrained test problems. Friedman tests have been performed as well as the multicomparison tests in order to obtain the statistical significance of the differences between the archiving methods. Table 6.1 - Table 6.4 show the box plots related to convergence and diversity metrics for unconstrained functions respectively together with the obtained p-values at the significance level  $\alpha = 0.05$ . The meaning of the metrics used is explained in section 5.1.1, page 121.

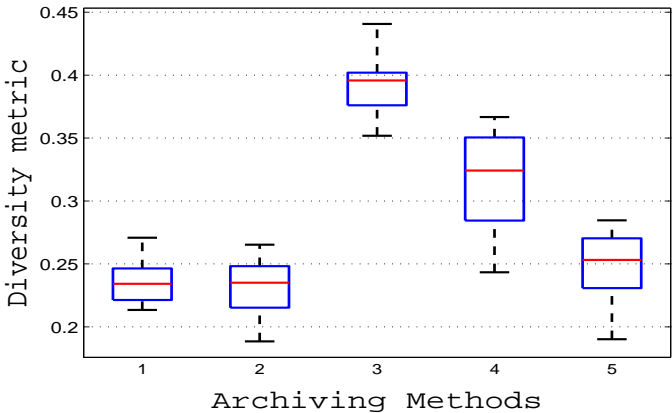
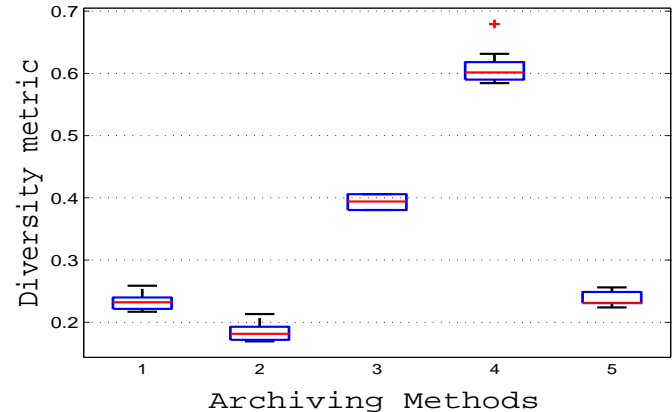
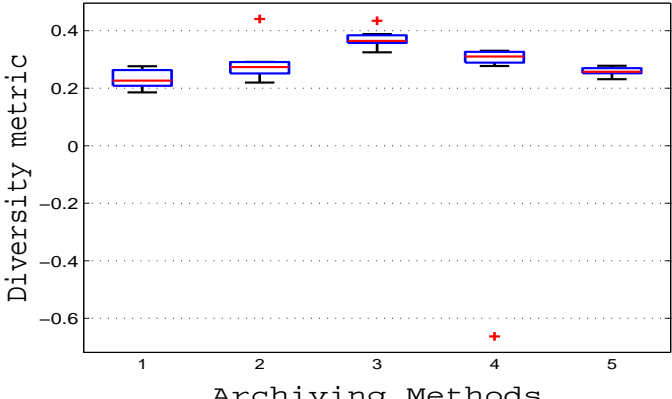
Convergence	p-value
 <p><b>Fonsceca</b></p>	$1.4557 * 10^{-7}$
 <p><b>Schaffer</b></p>	0.0008
 <p><b>ZDT1</b></p>	$2.4232 * 10^{-6}$

Tabel 6.1: Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to convergence results on unconstrained functions (FON, SCH, ZDT1) with five archiving methods numbered as 1. Unbounded, 2. Clustering, 3. Crowding 4. Maximin and 5. Redundancy removal

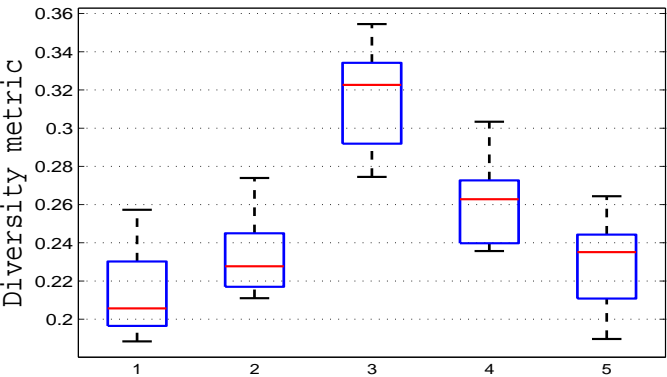
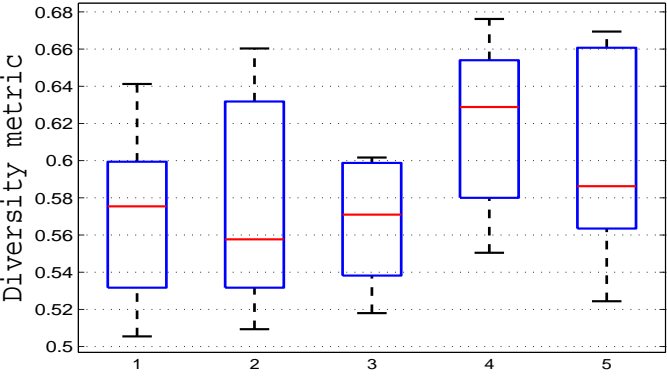
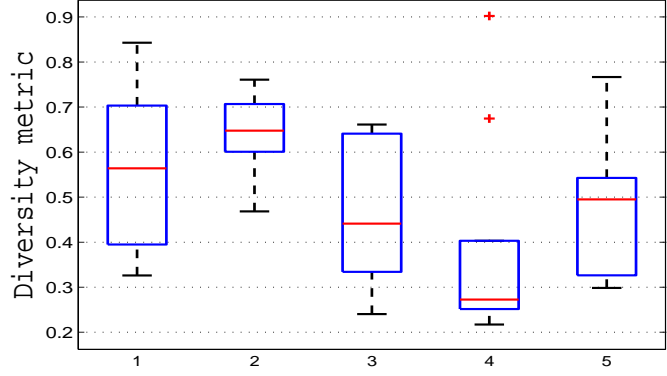
Convergence	p-value
<p><b>ZDT2</b></p>  <p>Convergence metric</p> <p>Archiving Methods</p>	0.0003
<p><b>ZDT3</b></p>  <p>Convergence metric</p> <p>Archiving Methods</p>	$2.1634 * 10^{-6}$
<p><b>ZDT6</b></p>  <p>Convergence metric</p> <p>Archiving Methods</p>	0.0378

Tabel 6.2: Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to convergence results on unconstrained functions (ZDT2, ZDT3, ZDT6) with five archiving methods numbered as 1. Unbounded, 2. Clustering, 3. Crowding 4. Maximin and 5. Redundancy removal



Diversity	p-value
<p style="text-align: center;"><b>Fonseca</b></p> 	$2.6840 * 10^{-6}$
<p style="text-align: center;"><b>Schaffer</b></p> 	$1.0788 * 10^{-7}$
<p style="text-align: center;"><b>ZDT1</b></p> 	$3.9539 * 10^{-5}$

Tabel 6.3: Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to diversity results on unconstrained functions (FON, SCH, ZDT1) with five archiving methods numbered as 1. Unbounded, 2. Clustering, 3. Crowding 4. Maximin and 5. Redundancy removal.

Diversity	p-value
<p style="text-align: center;"><b>ZDT2</b></p>  <p style="text-align: center;">Archiving Methods</p>	<p style="text-align: center;"><math>3.9539 * 10^{-5}</math></p>
<p style="text-align: center;"><b>ZDT3</b></p>  <p style="text-align: center;">Archiving Methods</p>	<p style="text-align: center;">0.4169</p>
<p style="text-align: center;"><b>ZDT6</b></p>  <p style="text-align: center;">Archiving Methods</p>	<p style="text-align: center;">0.0168</p>

Tabel 6.4: Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to diversity results on unconstrained functions (ZDT2, ZDT3, ZDT6) with five archiving methods numbered as 1. Unbounded, 2. Clustering, 3. Crowding 4. Maximin and 5. Redundancy removal.

We can see from these box plots that no variant has shown to be the best of all unconstrained problems from both convergence and diversity points of view. In terms of convergence, MOQPSO with the unbounded archive size strategy and maximin strategy exhibits better results in general compared with the other strategies. As expected, MOQPSO with the proposed redundancy removal archiving method has achieved intermediate results (better than clustering and crowding methods and less than unbounded and maximin methods) for all test problems except for ZDT3 where it presents the best result compared to all other strategies. The clustering and the crowding strategies alternate to occupy the fourth and the fifth position in the performance ordering. The p-values that have been recorded for the convergence metric in all unconstrained test functions and shown in Tables 6.1 and 6.2 present a significant difference between methods at the significance level  $\alpha = 0.05$ .

From the diversity point of view, it comes out that the unbounded archive size and the clustering strategies compete for the first and second ranks while maximin and crowding for the fourth and fifth ranks as shown in Tables 6.3 and 6.4. An exception can be clearly noticed for ZDT6 test function where maximin method presents the best performance followed by the crowding method. The redundancy removal method achieves intermediate results in general. This is very interesting because redundancy removal succeeded in achieving a good balance between convergence and diversity for almost all unconstrained functions with a remarkable reduction in CPU time when compared to the unbounded archive size strategy as shown in Figure 6.1. The p-values that have been recorded for the diversity metric in all unconstrained test functions and displayed in Tables 6.3 and 6.4 reveal that there is a significant difference between methods for all unconstrained test problems on the diversity metric at the significance level  $\alpha = 0.05$  except for ZDT3 test problem (p-value= 0.42).

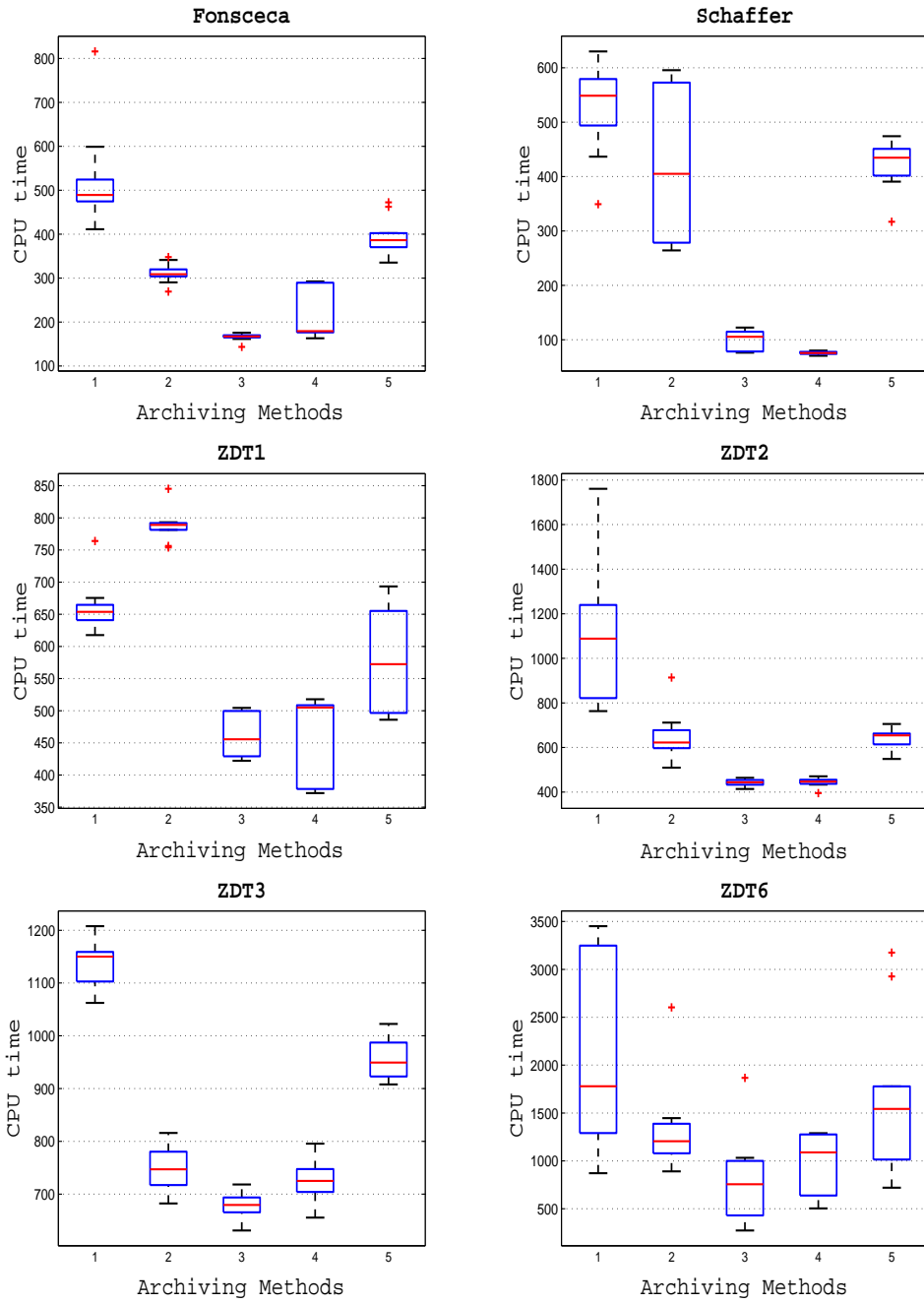
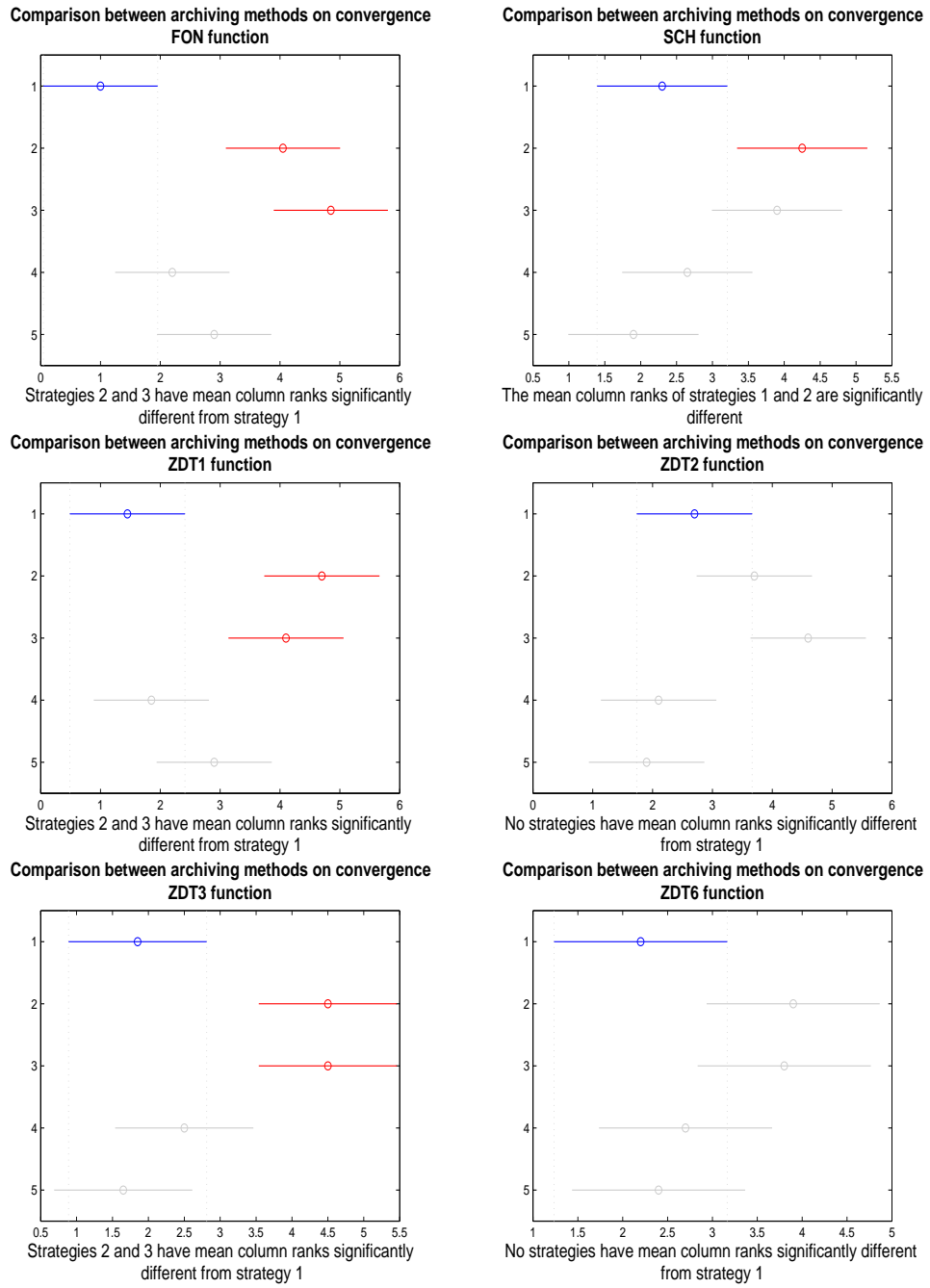


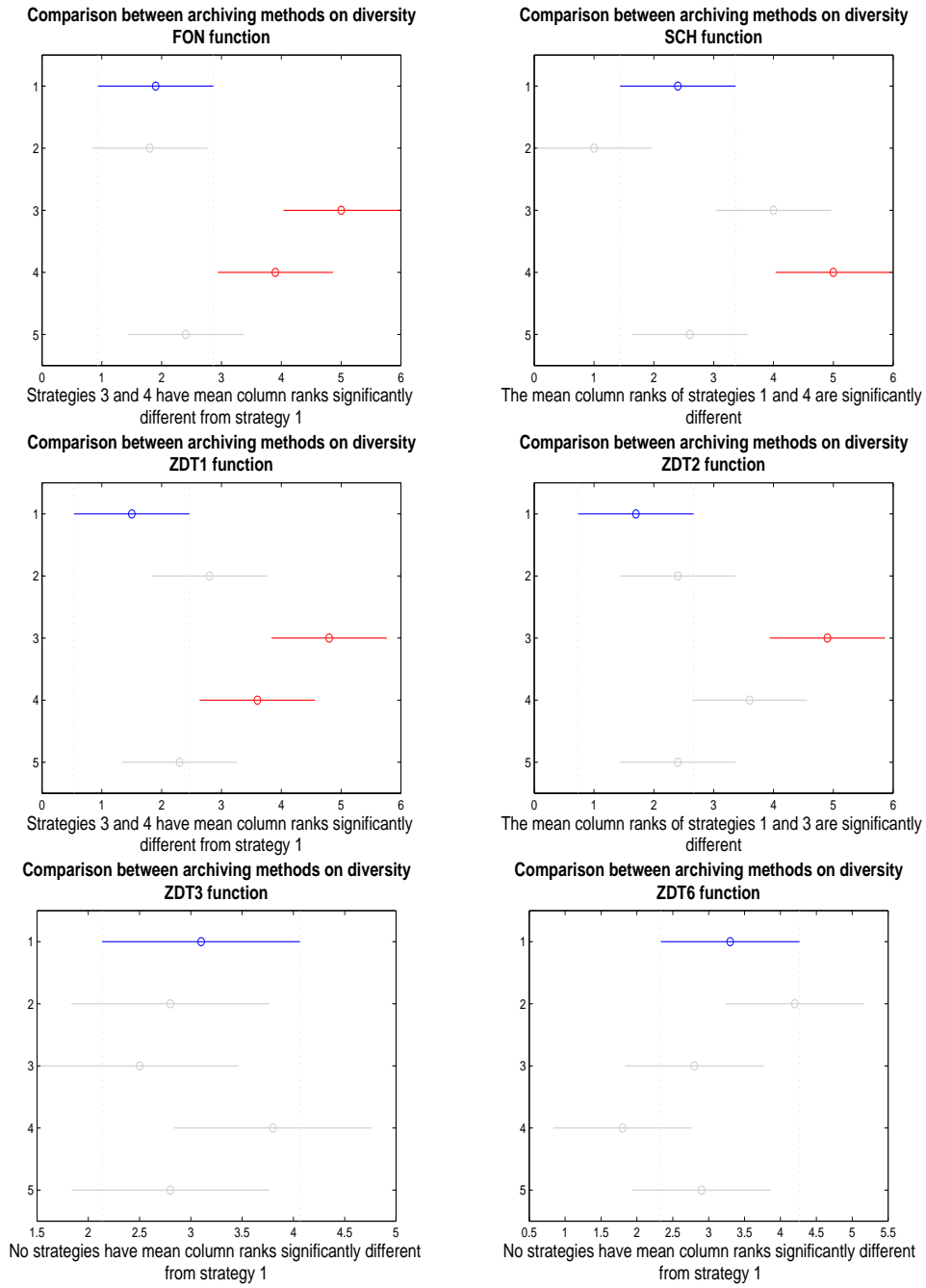
Figure 6.1: Box plots of CPU time for unconstrained functions with five archiving methods based on 1. Unbounded archive, 2. Clustering, 3. Crowding, 4. Maximin, and 5. Redundancy removal.

Analysis of the multicomparison graphs in Figures 6.2 and 6.3 has lead to the following observations: No significant difference has been found from both convergence and diversity points of view between redundancy removal and unbounded archive strategies, between redundancy removal strategy and maximin strategy except on diversity of SCH fronts and between redundancy removal strategy and clustering method except on convergence of SCH and ZDT3 fronts. Furthermore, the crowding archiving method exhibits no significant difference compared to the redundancy removal method in case of convergence on ZDT1 and ZDT6 fronts and in case of diversity on SCH, ZDT3 and ZDT6 fronts. In general, when considering the bounded archive size methods (clustering, crowding and maximin), MOQPSO achieves better convergence with the maximin method and better diversity with the clustering method. For the redundancy removal method, competitive results have been obtained when compared with the results of the unbounded method with an advantage of gain in CPU time over the unbounded archive size strategy.

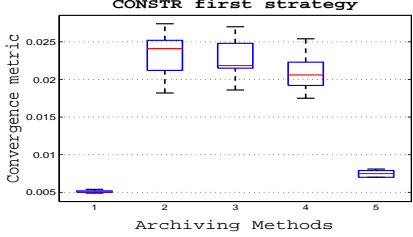
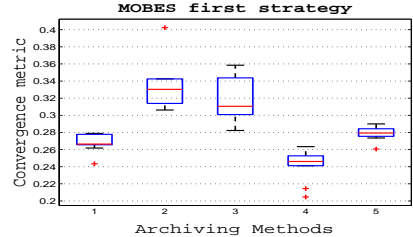
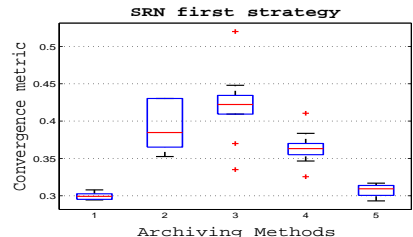
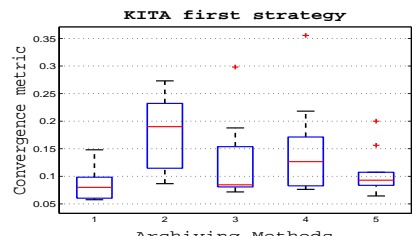
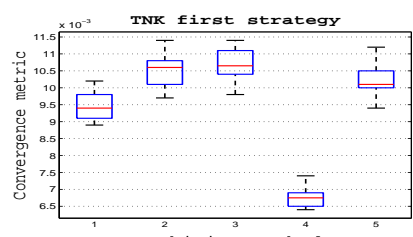
Regarding constrained problems, with the first constraint handling strategy it can be noticed from Table 6.5 that CMOQPSO with the unbounded archive method presents better convergence metric values in the case of CONSTR, SRN, and KITA test functions. Whereas, maximin method exhibits better performance in convergence metric on MOBES and TNK test functions. The redundancy removal method shows intermediate results in general when compared to the remaining archiving methods. Actually, it performs better than the maximin method in some cases such as CONSTR, SRN, and KITA functions. Based on the p-values recorded for the convergence metric and shown in Table 6.5, there is a significant difference between the archiving methods at the significance level  $\alpha = 0.05$  for all constrained test functions. According to the multicomparison test graphs shown in Figure 6.4, there is no significant difference between the unbounded archive and redundancy removal strategies for all constrained test problems.



Figur 6.2: Graphs of multicomparison tests of archiving methods based on Friedman statistics on convergence values for unconstrained functions. Overlapping intervals indicate no significant difference



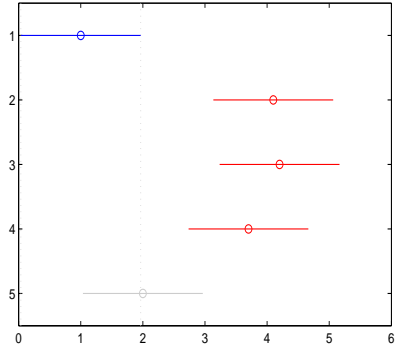
Figur 6.3: Graphs of multicomparison tests of archiving methods based on Friedman statistics on diversity values for unconstrained functions. Overlapping intervals indicate no significant difference

Convergence	p-value
 <p>CONSTR first strategy</p>	$1.4697 * 10^{-6}$
 <p>MOBES first strategy</p>	$2.5827 * 10^{-7}$
 <p>SRN first strategy</p>	$8.3460 * 10^{-7}$
 <p>KITA first strategy</p>	0.0068
 <p>TNK first strategy</p>	$4.4333 * 10^{-6}$

Tabel 6.5: Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to convergence results for constrained functions with first constraint strategy on five archiving methods based on 1. Unbounded, 2. Clustering, 3. Crowding, 4. Maximin, and 5. Redundancy removal.

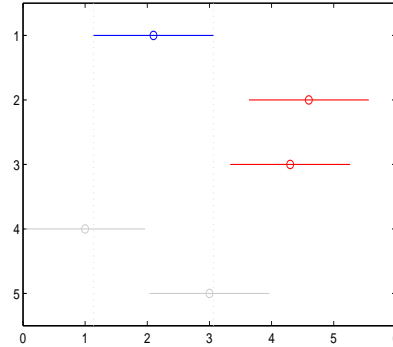


Comparison between archiving methods on convergence  
CONSTR – constraint strategy 1



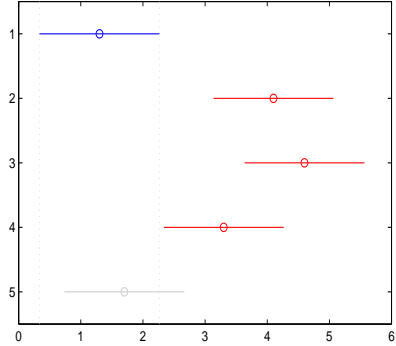
Archiving methods 2, 3 and 4 have mean column ranks significantly different from method 1

Comparison between archiving methods on convergence  
MOBES – constraint strategy 1



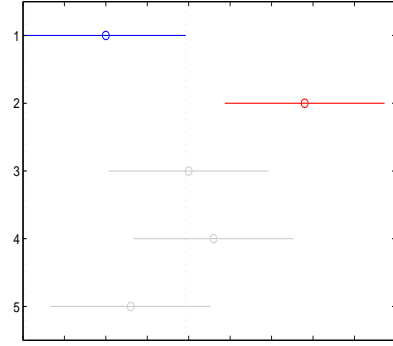
Archiving methods 2 and 3 have mean column ranks significantly different from method 1

Comparison between archiving methods on convergence  
SRN – constraint strategy 1



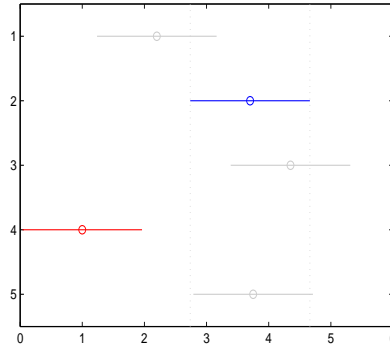
Archiving methods 2, 3 and 4 have mean column ranks significantly different from method 1

Comparison between archiving methods on convergence  
KITA – constraint strategy 1



Archiving methods 1 and 2 are significantly different

Comparison between archiving methods on convergence  
TNK – constraint strategy 1

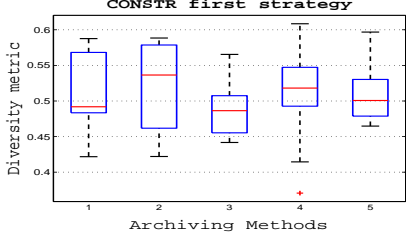
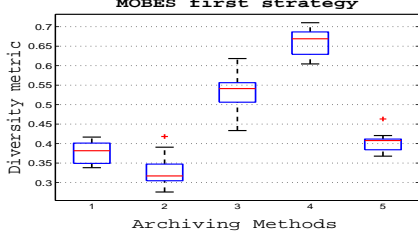
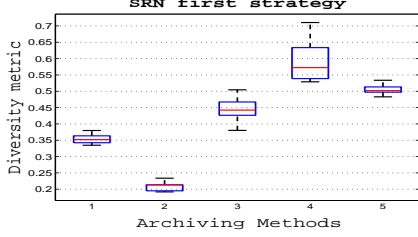
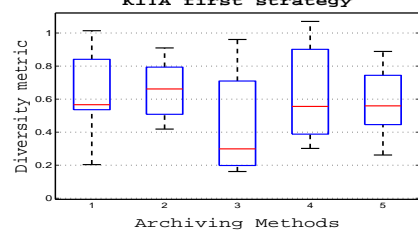
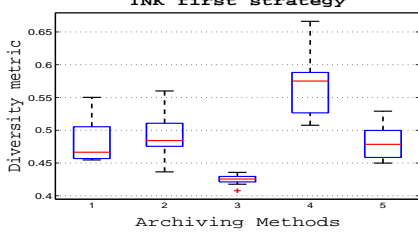


The mean column ranks of archiving methods 2 and 4 are significantly different

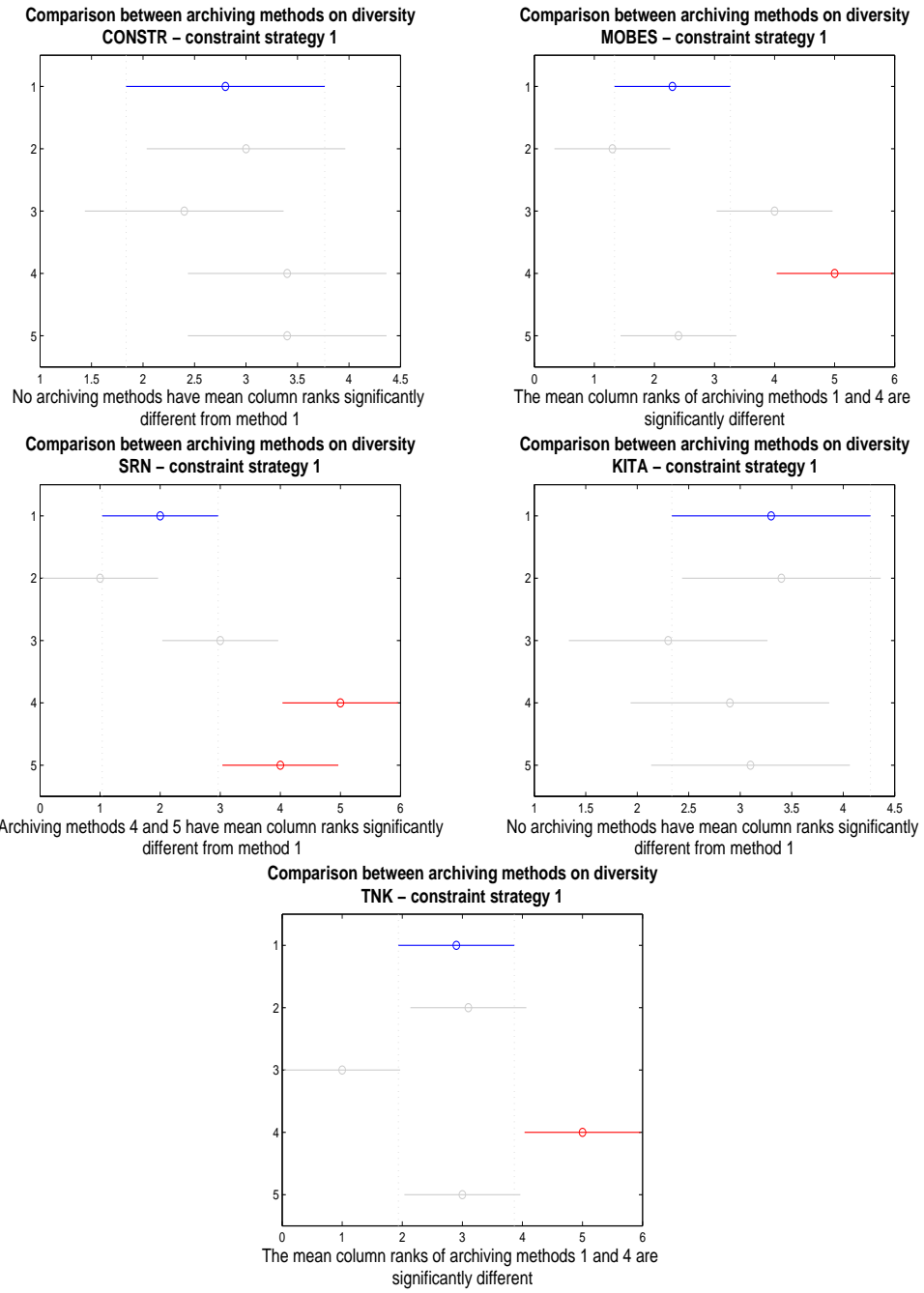
Figur 6.4: Graphs of multicomparison tests of archiving methods based on Friedman statistics on convergence values for the first constraint strategy. Overlapping intervals indicate no significant difference.

For the diversity metric, Table 6.6 presents the diversity values of the constrained problems with the first strategy. Best results have been recorded for both the clustering method (on MOBES and SRN) and the crowding method (on CONSTR, KITA, and TNK). The redundancy removal exhibits intermediate results in all cases. There is also no significant difference between the unbounded archive and redundancy removal method based on the multicomparison test graphs in Figure 6.5. Furthermore, the redundancy removal method requires less computational time than the unbounded archive method. This can be clearly seen in the box plots of the CPU time in Figure 6.6.

For the second strategy, we can see from Table 6.7 that MOQPSO with the unbounded archive method exhibits better convergence results especially in the case of SRN, CONSTR, and TNK functions. The maximin method shows good convergence in MOBES function whereas the redundancy removal method achieves in general intermediate results compared to the other archiving methods. For example, it performs better than maximin in SRN and KITA functions. However, no significant difference has been found between unbounded archive strategy and redundancy removal strategy as shown by the multicomparison test graphs in Figure 6.7 except on convergence of CONSTR front. From diversity point of view and according to the box plots shown in Table 6.8, CMOQPSO with the redundancy removal method achieves intermediate results in all cases. Best diversity results have been achieved with either clustering (SRN, KITA, and MOBES functions) or crowding (CONSTR function). No significant differences between redundancy removal and unbounded archive strategies have been detected except on the SRN function as shown in Figure 6.8. However, from the box plots of CPU time in Figure 6.9 we can see that the redundancy removal method requires a CPU time which is up to three times less than that of the unbounded archive method with a slight gain in convergence quality achieved by this latter.

Diversity	p-value
	0.5781
	$4.7341 * 10^{-7}$
	$4.3284 * 10^{-8}$
	0.5512
	$1.8424 * 10^{-6}$

Tabel 6.6: Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to diversity results for constrained functions with first constraint strategy on five archiving methods based on 1. Unbounded, 2. Clustering, 3. Crowding, 4. Maximin, and 5. Redundancy removal.



Figur 6.5: Graphs of multicomparison tests of archiving methods based on Friedman statistics on diversity values for the first constraint strategy. Overlapping intervals indicate no significant difference.

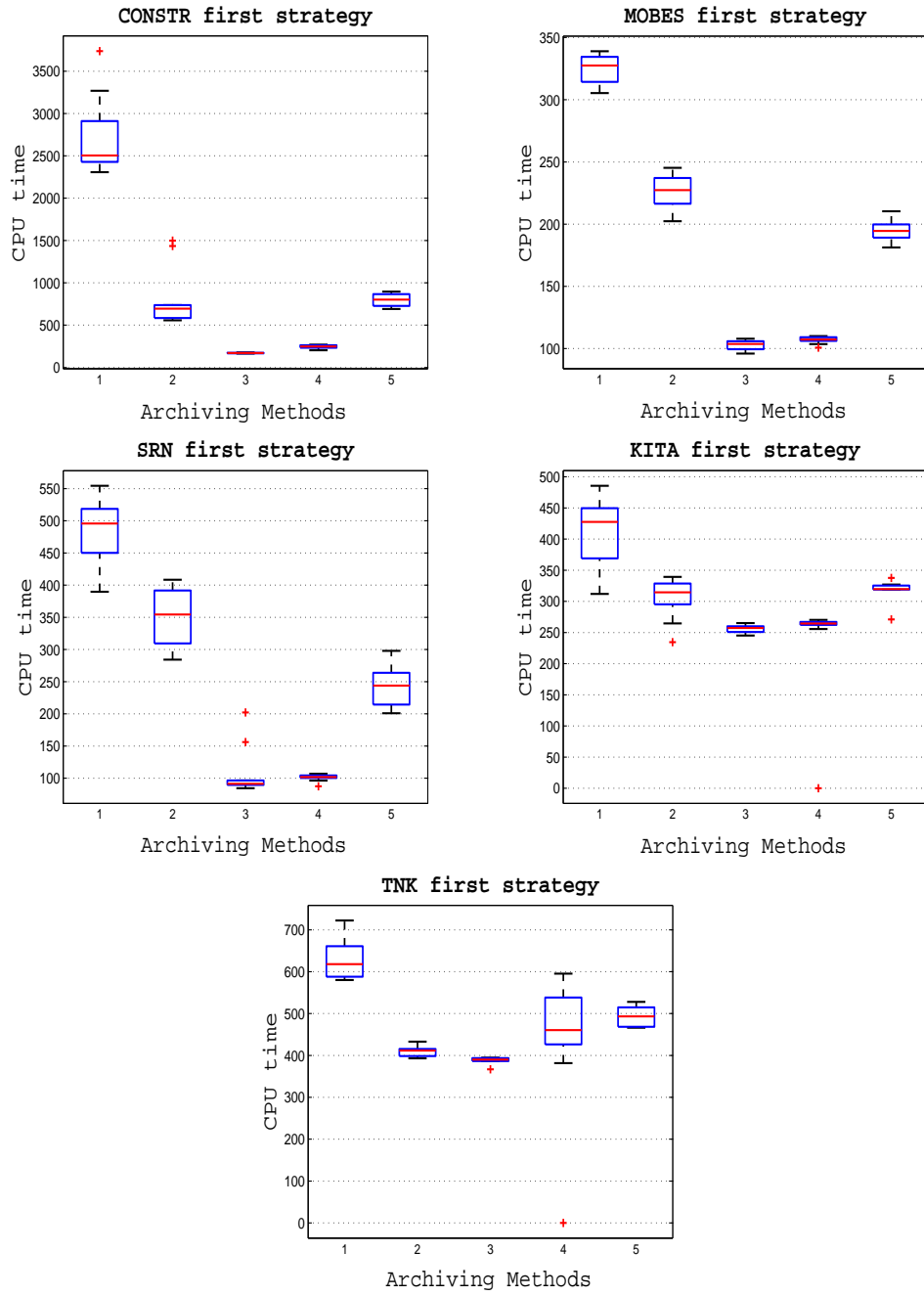
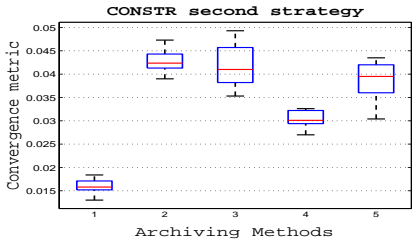
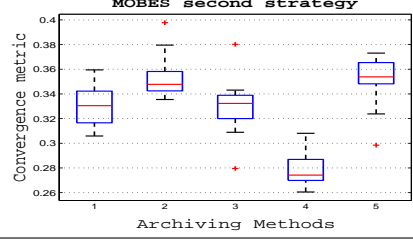
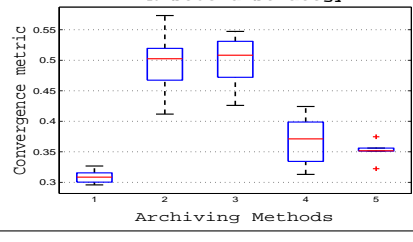
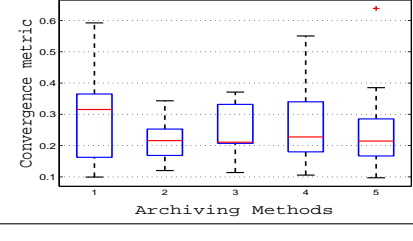
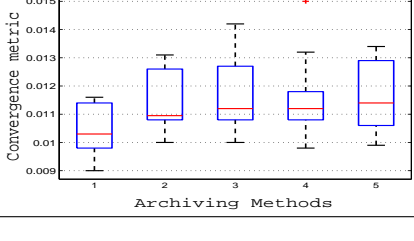
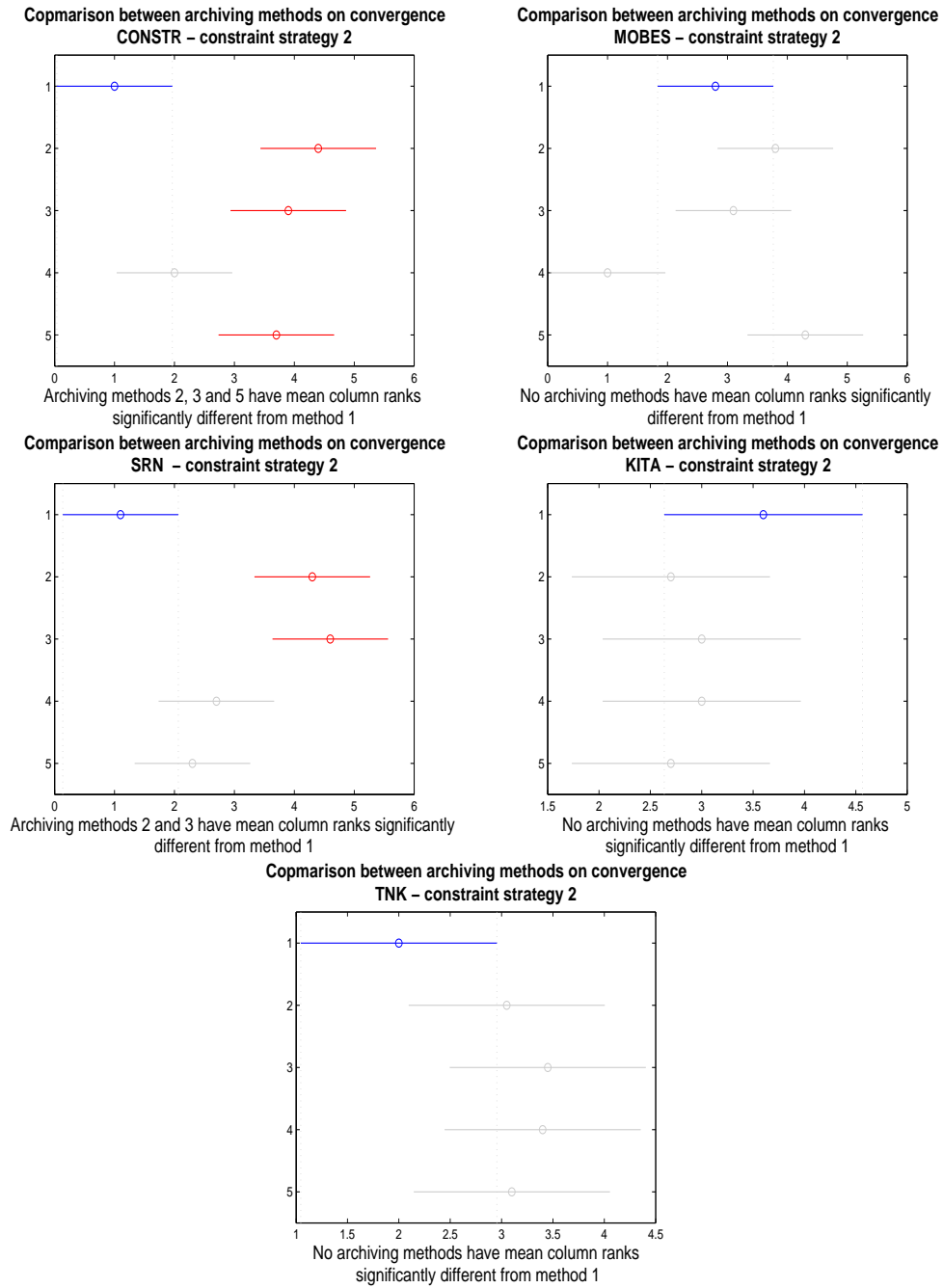


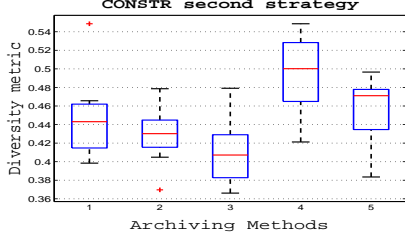
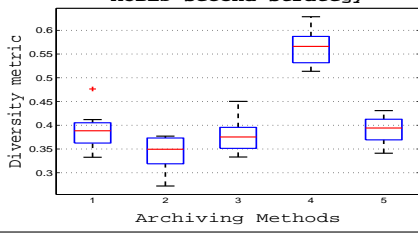
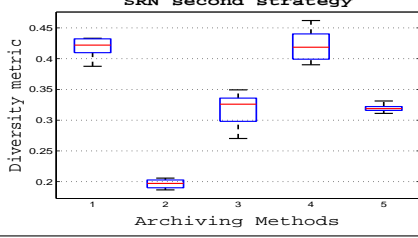
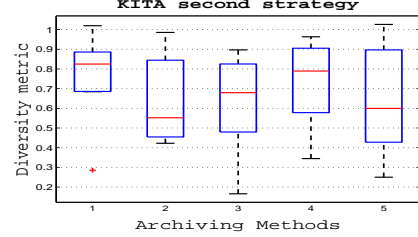
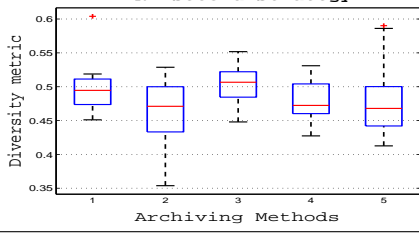
Figure 6.6: Box plots of CPU time for constrained functions with first constraint strategy on five archiving methods based on 1. Unbounded, 2. Clustering, 3. Crowding, 4. Maximin, and 5. Redundancy removal.

Convergence	p-value
 <p>CONSTR second strategy</p>	$1.1722 * 10^{-6}$
 <p>MOBES second strategy</p>	$3.9539 * 10^{-5}$
 <p>SRN second strategy</p>	$8.3460 * 10^{-7}$
 <p>KITA second strategy</p>	0.7064
 <p>TNK second strategy</p>	0.2300

Tabel 6.7: Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to convergence results for constrained functions with second constraint strategy on five archiving methods based on 1. Unbounded, 2. Clustering, 3. Crowding, 4. Maximin, and 5. Redundancy removal.



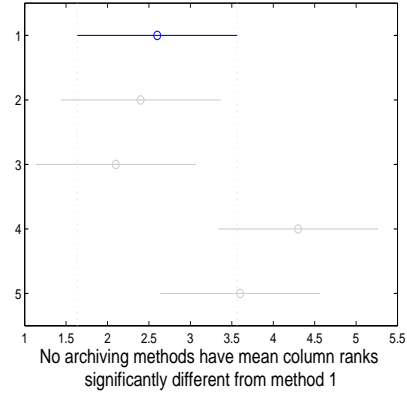
Figur 6.7: Graphs of multicomparison tests of archiving methods based on Friedman statistics on convergence values for the second constraint strategy. Overlapping intervals indicate no significant difference.

Diversity	p-value
 <p>CONSTR second strategy</p>	0.0090
 <p>MOBES second strategy</p>	0.0001
 <p>SRN second strategy</p>	$2.8937 * 10^{-7}$
 <p>KITA second strategy</p>	0.5918
 <p>TNK second strategy</p>	0.5918

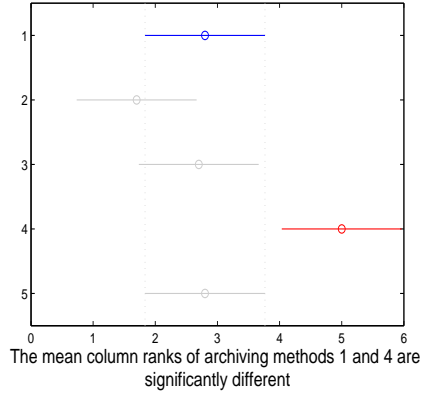
Tabel 6.8: Box plots and p-values with respect to ( $p < \alpha = 0.05$ ) related to diversity results for constrained functions with second constraint strategy on five archiving methods based on 1. Unbounded, 2. Clustering, 3. Crowding, 4. Maximin, and 5. Redundancy removal.



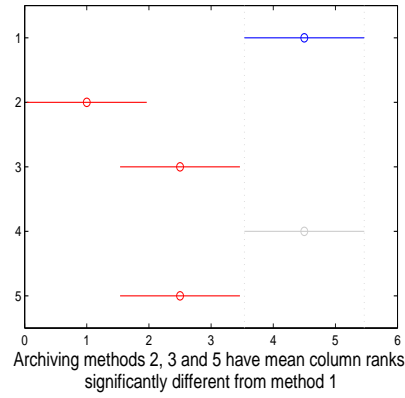
Comparison between archiving methods on diversity  
CONSTR – constraint strategy 2



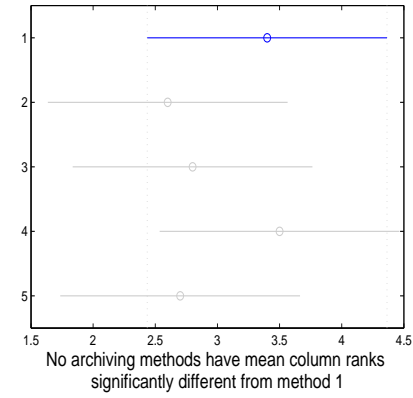
Comparison between archiving methods on diversity  
MOBES – constraint strategy 2



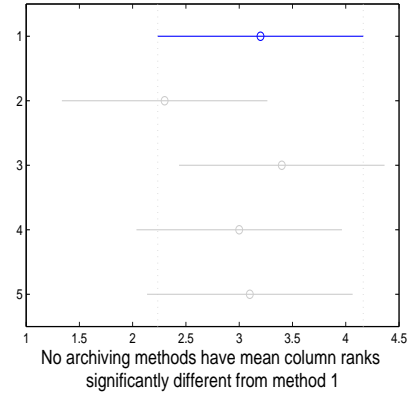
Comparison between archiving methods on diversity  
SRN – constraint strategy 2



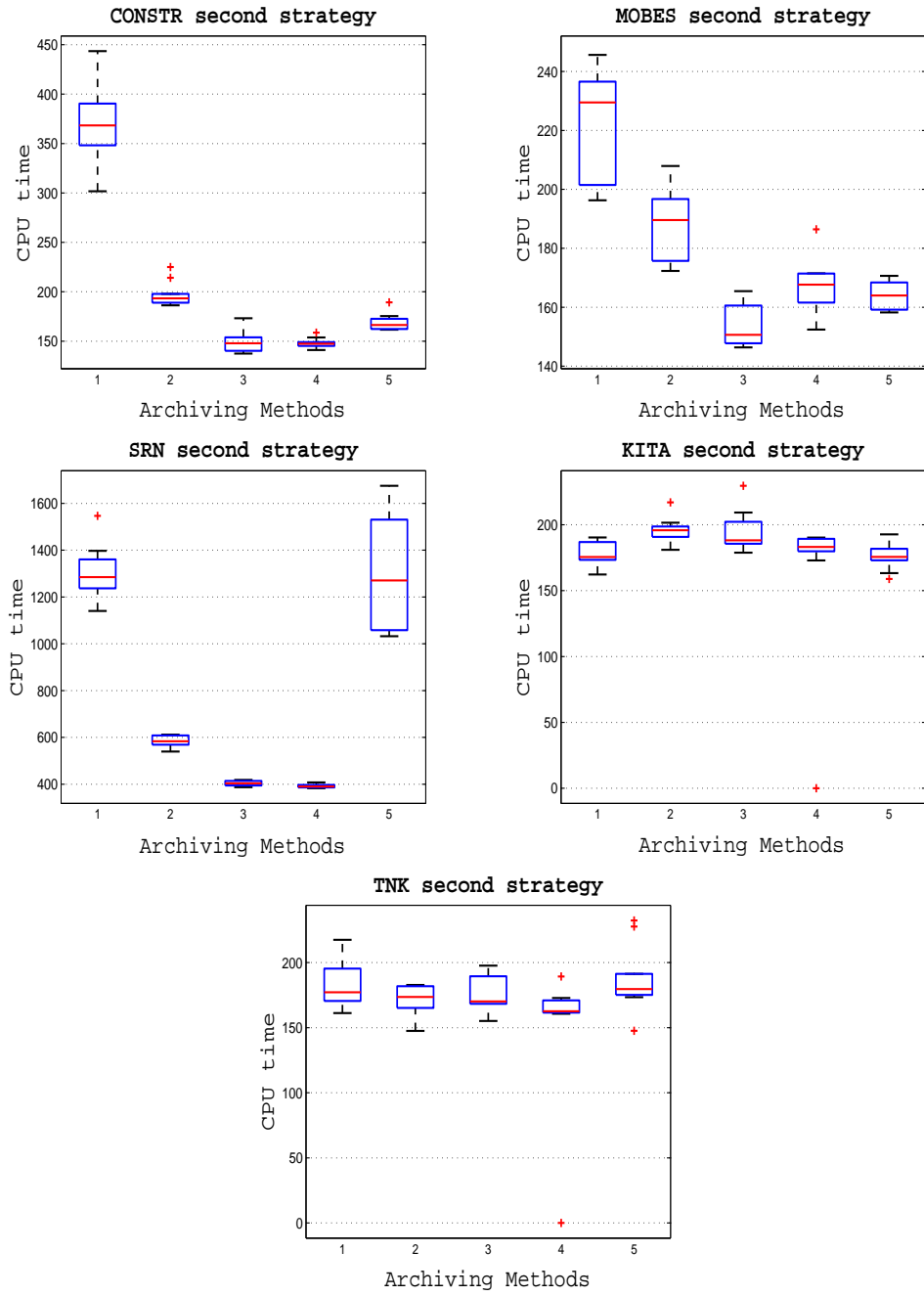
Comparison between archiving methods on diversity  
KITA – constraint strategy 2



Comparison between archiving methods on diversity  
TNK – constraint strategy 2



Figur 6.8: Graphs of multicomparison tests of archiving methods based on Friedman statistics on diversity values for the second constraint strategy. Overlapping intervals indicate no significant difference.



Figur 6.9: Box plots of CPU time for constrained functions with second constraint strategy on five archiving methods based on 1. Unbounded, 2. Clustering, 3. Crowding, 4. Maximin, and 5. Redundancy removal.

Generally, the first constraint handling strategy performs better than the second constraint handling strategy with respect to the convergence metric for all test problems and with respect to the diversity metric for most of the test functions. However, the time complexity of the first constraint strategy is higher than the second constraint strategy. To recapitulate, from the above results the conclusion is that the proposed redundancy removal strategy helps archiving the same quality of solutions as the unbounded archive strategy with a considerable gain in CPU time for both constrained and unconstrained test problems.

The growth of the archive size through iterations has been also recorded. Figures 6.10, 6.11, and 6.12 show the archive size growth for both unconstrained and constrained functions respectively. As it can be seen from these figures, the unbounded archive size method induces a rapid growth while with the clustering, maximin and crowding archiving methods the archive size grows till the limit is reached. The redundancy removal strategy leads to a zigzag growth that lies in general between those of the unbounded archive strategy on one hand and the other bounded methods on the other hand.

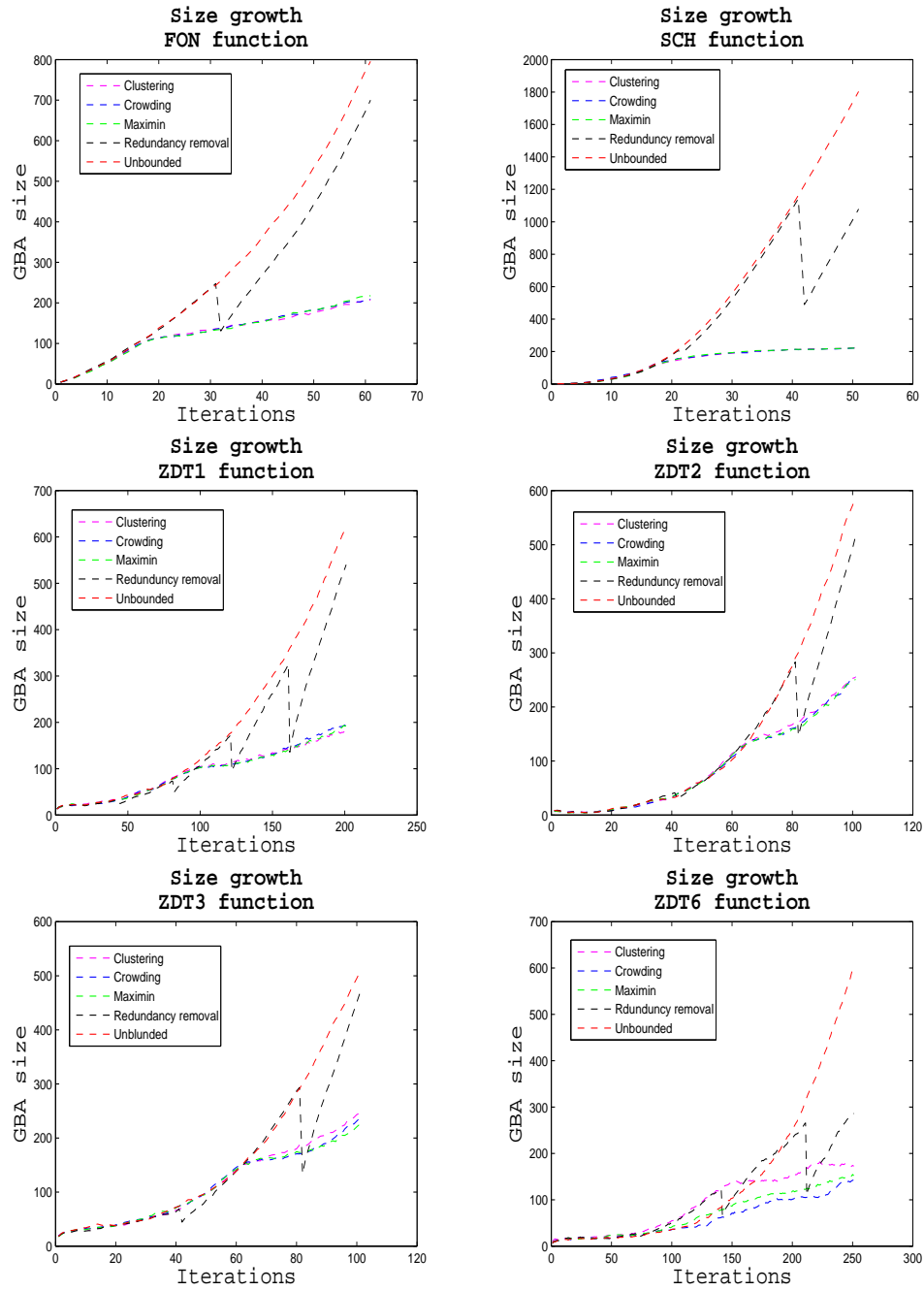


Figure 6.10: Archive size growth of the five archiving methods for unconstrained functions.

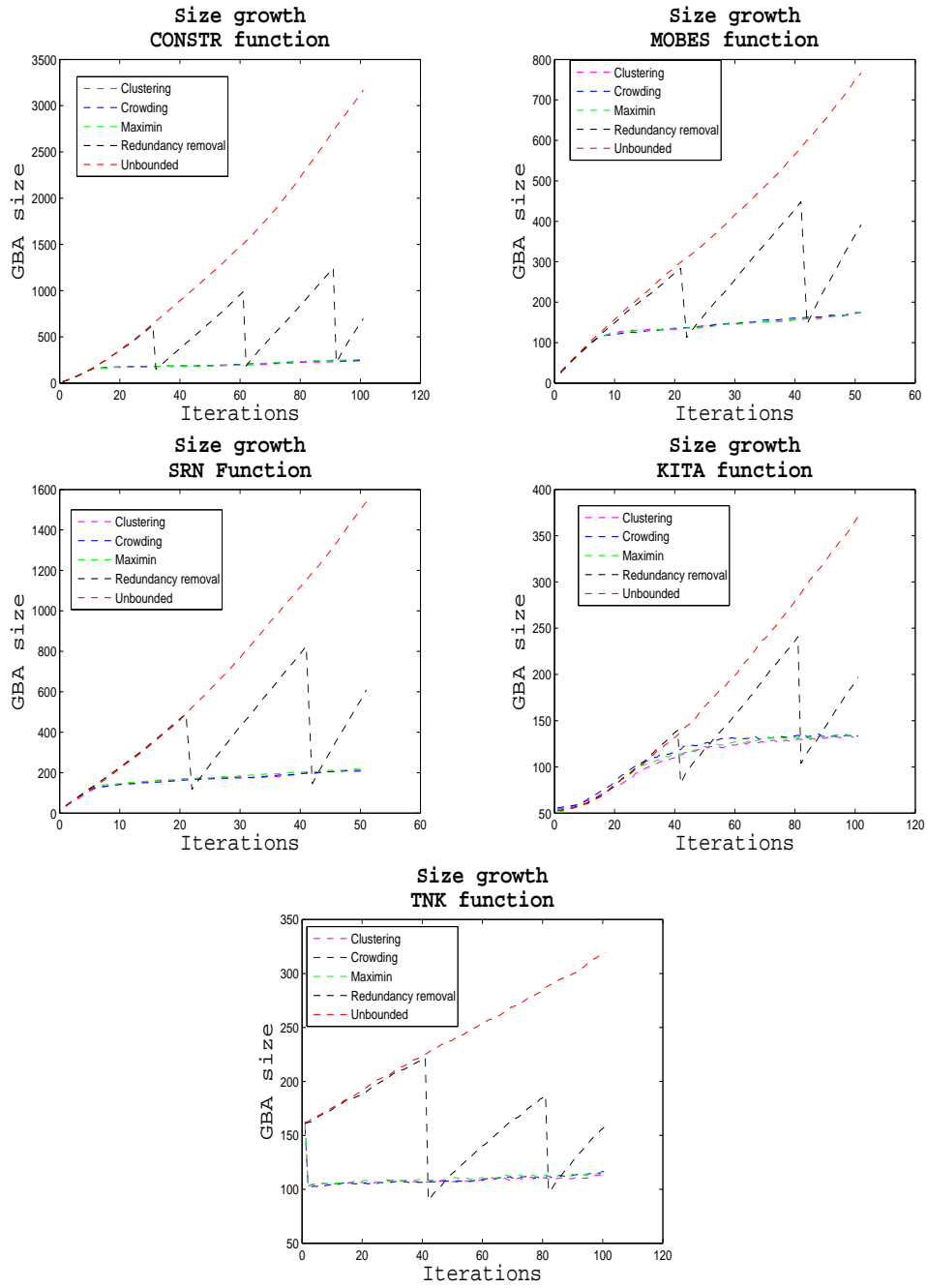


Figure 6.11: Archive size growth of the five archiving methods for constrained functions with the first constraint handling strategy.

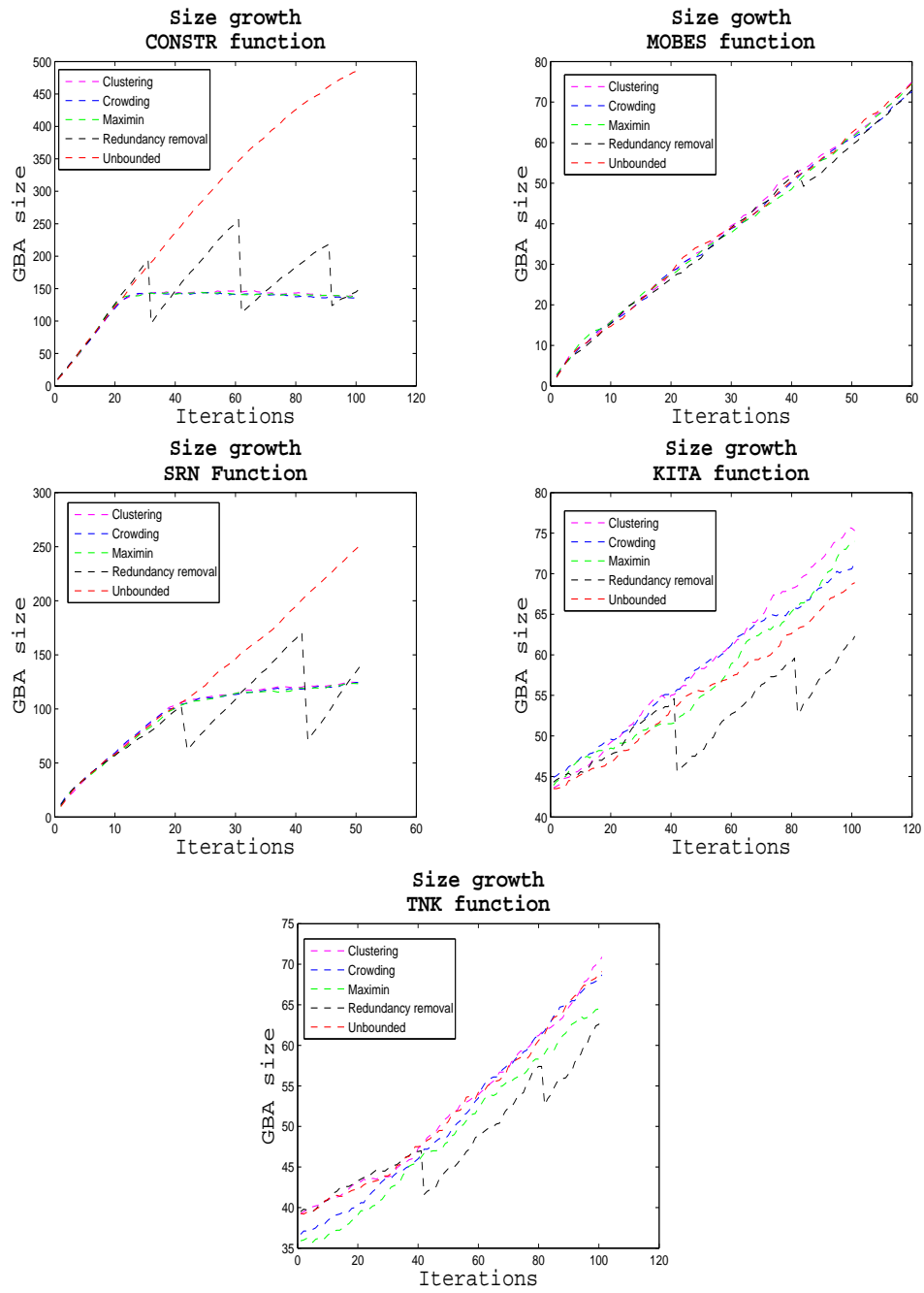


Figure 6.12: Archive size growth of five archiving methods for constrained functions with the second constraint handling strategy.

## 6.4 Summary

For what we can gather from the experiments performed for the archiving methods:

- For unconstrained test problems:
  - MOQPSO with the unbounded archive size and the maximin strategies exhibits better convergence results in general when compared with the other archiving strategies.
  - As expected, MOQPSO with the proposed redundancy removal method achieves intermediate convergence results. It performs relatively better than clustering and crowding methods and slightly less than unbounded and maximin methods for all test problems.
  - MOQPSO with the unbounded archive size and the clustering strategies records better diversity results when compared with the rest of the archiving methods. The redundancy removal strategy achieves intermediate results in general. It performs better than maximin and crowding methods and slightly less than unbounded archive and clustering methods.
  - For the bounded archive size methods (clustering, crowding and maximin), MOQPSO achieves better convergence with the maximin method and better diversity with the clustering method.
  - It is observed that MOQPSO with the proposed redundancy removal method succeeded in maintaining a good balance between convergence and diversity for almost all unconstrained functions. It actually shows competitive results when compared against the unbounded archive method with a remarkable saving in CPU time over the unbounded archive method and no significant difference between the two methods.

- For constrained test problems:
  - In both constraint handling strategies, CMOQPSO obtains better convergence results with the unbounded archive and the maximin methods and it obtains better diversity results with the clustering and the crowding methods.
  - CMOQPSO with the proposed removal redundancy method presents intermediate results with respect to convergence and diversity metrics when compared with the other archiving methods.
  - The redundancy removal method succeeded in obtaining very competitive results when compared against the unbounded archive method with an advantage of gain in CPU time over the unbounded archive size method and no significant difference between both methods.
  - For the first constraint handling strategy with the bounded archive size methods (clustering, crowding and maximin), better convergence results have been achieved with the maximin method and better diversity results have been obtained with both the clustering and the crowding methods.
  - For the second constraint handling strategy with the bounded archive size methods (clustering, crowding and maximin), better convergence results have been achieved with the maximin method in most of the test functions and better diversity results have been obtained with the clustering method in most of the test functions.
  - The first strategy performs better than the second strategy in obtaining best convergence in all of the test functions and better diversity results in most of the test functions.
  - The first strategy requires more computational time than the second constraint strategy.



Generally, the crowding based archiving strategy has a better time complexity than the redundancy removal method and the remaining bounded archiving methods, namely clustering and maximin. However, its performance is inferior to these methods in terms of convergence and diversity. The redundancy removal method scales in the same way as the maximin and the clustering methods with respect to the archive size and the number of objectives.

Overall, it is observed that the proposed redundancy removal archiving method successfully performed against the other archiving methods (unbounded and bounded). The experimental results show that the main objectives of the proposed method have been met in that it takes the advantage of the unbounded archive size method in achieving good convergence to the Pareto front and maintaining diverse solutions but with an effective computational time without the need to limit the archive size as required by the other bounded archiving methods.

---

### MOQPSO-Clust: An Application of MOQPSO to Clustering

---

In the previous chapters we have built a novel framework for multi-objective optimization and have demonstrated its effectiveness on a battery of benchmark test functions. We are now in the position to apply our methodology to real-world application domains. This chapter demonstrates this in application to clustering – an important tool in many fields such as exploratory data mining and pattern recognition.<sup>1</sup> Clustering consists of organizing a large data set into groups of objects that are more similar to each other than to those in other groups. Despite its use for over three decades, it is still subject to a lot of controversy and remains a challenging task. In this chapter, we demonstrate the application of the proposed MOQPSO in cluster analysis problems. We cast clustering as a Pareto based multi-objective optimization problem which is handled using a quantum behaved particle swarm optimization algorithm. The search process is carried out over the space of cluster centroids with the aim to find out partitions that optimize two objectives

---

<sup>1</sup>A shorter version of the work in this chapter has been published in the following: Heyam Al-Baity, Souham Meshoul, Ata Kaban and Lilac Alsafadi. Quantum Behaved Particle Swarm Optimization for Data Clustering with Multiple Objectives. IEEE Sixth International Conference on Soft Computing and Pattern Recognition, (IEEE SOCPAR), pp. 215-220, 2014.

simultaneously, namely compactness and connectivity.

## 7.1 Overview on Data Clustering

Recent advances in information technology have fostered the creation of large quantities of data. These datasets are more often unstructured and therefore difficult to analyze. Data clustering, also known as cluster analysis, is the process of partitioning a dataset into meaningful groups (clusters), such that the objects in the same group are similar to each other and dissimilar to the objects in the other groups. It is considered as an unsupervised classification problem where classes are not known in advance [42][62]. Clustering is one of the most important techniques in data mining and has been applied to many interesting applications such as image processing [59], machine learning [18], bioinformatics [7], and document classification and web mining [72]. Clustering is a data analysis tool that is used to reveal hidden patterns and organize the data in a way that allows the users to gain some insight about the content of the data and summarize data into useful information [58].

The main objective of clustering is to segment large data into meaningful clusters so that the intra-cluster homogeneity and the inter-cluster heterogeneity are both maximized [54][58]. There are many clustering methods that have been proposed in the literature over the past decades of which we present a short overview in the sequel, for completeness. According to [59], the clustering techniques can be broadly classified into two types:

- **Hierarchical (nested) clustering**

This class aims at decomposing data iteratively into nested clusters by using either the agglomerative (bottom-up or singleton clusters) strategy or the divisive (top-down) strategy. The agglomerative approach starts with each data point in a separate cluster (singletons), then recursively merges them into bigger clusters based

on their similarities until a termination criterion is satisfied. The divisive method is the opposite of the agglomerative method. It begins with the entire dataset as one cluster and recursively divides it into smaller clusters until a termination criterion is reached. The result of hierarchical clustering is a tree-like structure – that is a graph called a dendrogram [59]. Single-link distance and complete-link distance are the most known algorithms within this category. They both compute the distance between clusters (i.e. inter-cluster linkage metric). The single-link distance, also called minimum distance, is the shortest distance between any two objects from two clusters and hence maximum of the similarity. On the other hand, complete-link distance, also called maximum distance, is the farthest distance between two objects from two clusters and hence minimum of the similarity [58][2].

- **Partitional (unnested) clustering**

This category partitions the data at once into a predefined number of non overlapping clusters (K) based on an objective function [59]. K-means is the most popular algorithm within this class of methods. It begins by initializing the cluster centres or centroids. A centroid is the mean of all objects in the cluster. Then, each object is assigned to the cluster with the closest centroid. In order to improve the clustering process, the cluster centroids are updated iteratively and the objects are reassigned to the new clusters. The algorithm ends when the cluster centroids stop changing. The aim of this method is to minimize the dissimilarity between objects and their cluster centres [58][42]. K-medoid is a variant of K-means algorithm. In this clustering technique, medoid is the object used to represent a cluster instead of the centroid. Medoid is the object closest to the cluster centre [58][71].

PAM (Partitioning Around Medoid) is a K-medoid based clustering algorithm that attempts to select the objects (medoids) for each cluster at first. Then the remaining non-selected objects are assigned to the cluster with the closest medoids [42][58].

Many other clustering techniques are available in the literature. Interested readers can refer to [116] for more details on clustering methods.

## 7.2 Why Multi-objective Clustering

The conventional clustering algorithms suffer from several problems. Their main disadvantage is that they attempt to optimize just a single clustering criterion. Such a single criterion may not be able to capture the intended notion of clusters given the diverse characteristics of the datasets [54]. Additionally, the quality of clustering resulting from partitional clustering algorithms depends highly on the initial settings of the centroids which may lead to locally optimal partitions. A common solution to the latter problem is to perform multiple runs of the algorithm with different initial centroids and then select the best partitioning results as the final clustering solution. However, this approach is not effective when dealing with a large dataset and a large number of clusters [71]. Another drawback common to the clustering techniques is the difficulty of choosing the right number of clusters in the data [71].

## 7.3 Clustering as a MOP

In order to overcome the problems in the above section and to obtain a good and meaningful clustering, global search optimization techniques such as Genetic algorithms (GAs) and Particle Swarm Optimization (PSO) can be employed to explore the search space and achieve better quality solutions. Moreover, the clustering solutions should be assessed from different aspects or different validity criteria, rather than a single aspect. Therefore, tackling the clustering problem as a truly multi-objective optimization problem would be a promising attempt in order to improve the quality of the final clustering solutions and to obtain a set of trade off solutions in a single run via Pareto based multi-objective optimization [2][71]. In the following section, we provide a brief review of multi-objective

nature inspired algorithms for clustering.

### 7.3.1 Related Work

Due to the importance of clustering in many fields, a large number of algorithms have been proposed in the literature to solve clustering problems. Recently, nature inspired algorithms such as Genetic Algorithms (GAs), Simulated Annealing (SA), and Particle Swarm Optimization (PSO) have been successfully applied to solve data clustering problems in a multi-objective context. For example, J. Handle and J. Knowles [50] proposed a multi-objective clustering with automatic K-determination called (MOCK). The algorithm consists of two phases. In the initial clustering phase, two clustering objectives are optimized namely, compactness and connectedness using the Pareto Envelope based Selection Algorithm (PESAI) [29]. In the second model selection phase, the quality of the set of partitioning solutions obtained from the first phase is assessed by an automated selection model called Gap statistic. This model then selects the final clustering solution and implicitly estimates the number of clusters.

H.Ali et al.[4] proposed a multi-objective PSO (MOPSO) based clustering algorithm for mobile ad hoc networks (MANET). Their aim is to find the optimal number of clusters and to select the cluster centroid for each cluster which can make the network energy efficient and hence increase its lifetime. Results show the effectiveness of the proposed approach when compared with two other clustering algorithms.

Sanghamitra Bandyopadhyay et al. [10] proposed a multi-objective fuzzy clustering algorithm that is based on NSGAII. The Xie-Beni (XB) index [115] and  $J_m$  measure [14] have been selected as the two objectives to be optimized simultaneously. The  $J_m$  computes the global intra-cluster variance over all clusters. The lower the values of  $J_m$ , the

better the partitioning solution. The XB index is the result of the division of the global cluster variance which is similar to  $J_m$  by the distance between the two nearest clusters. Therefore, the XB measure is optimized (minimized) when the global cluster variation is minimized and the distance between the two closest clusters is maximized.

## 7.4 The Proposed MOQPSO for Clustering (MOQPSO-Clust)

In this section, we adopt MOQPSO to solve the data clustering problem in order to find the possible partitions of various datasets according to multiple objectives, namely compactness and connectivity. The objective of this work is twofold. We demonstrate on one hand the ability of MOQPSO to handle the clustering problem, and on the other hand the ability of our multi-objective clustering to obtain meaningful trade-off partitions.

### 7.4.1 Problem formulation

The multi-objective clustering problem can be defined as follows [71][50]:

Given a dataset  $E$  consisting of  $n$  points,  $E = \{e_1, e_2, \dots, e_n\}$ , multi-objective clustering is the task that consists in finding the set of non-dominated partitions or clusters  $\vec{C}^* = (c_1^*, c_2^*, \dots, c_k^*)$  of  $E$  that optimizes (minimizes or maximizes) a vector of objective functions  $\vec{F}_t$ ,  $t = 1, \dots, m$ , which measures the quality of a partition using  $m$  objective functions. Each  $c_i$  denotes a cluster and  $k$  is the number of clusters. More formally, the problem can be formulated as follows (in case the objectives should be minimized):

Given  $E = \{e_1, e_2, \dots, e_n\}$ , find the partition  $\vec{C}^* = (c_1^*, c_2^*, \dots, c_k^*)$  of  $E$  such that

$$\vec{C}^* = \operatorname{argmin}_{C \in \Omega} (\vec{F}_t(C) = (f_1(C), f_2(C), \dots, f_m(C)))$$

subject to:

- $\forall i \quad c_i \neq \emptyset$
- $\forall i, j \quad c_i \cap c_j = \emptyset \quad \text{for } i \neq j$
- $c_1 \cup c_2 \cup \dots c_k = E$

Where  $\Omega$  is the set of all potential clustering solutions or partitions. Therefore the decision search space is defined by the space of partitions.

### 7.4.2 Cluster Encoding

In order to solve clustering with multi-objective QPSO, a suitable way of particles encoding should be adopted in order to represent the potential clustering solutions. As we propose an approach that is based on the principle of partitional clustering, each particle position is related to the centroids of the clusters. In other words, the potential solution in the proposed MOQPSO clustering algorithm represents a partition or clustering which is given by a set of cluster centroids. Therefore, the search is performed in the space of centroids. For instance, assume we have a small dataset of 5 points defined by three attributes and number of clusters  $k=2$  as shown in Figure 7.1. The representation of the potential partitions of this dataset is shown in Figure 7.2. The problem dimension in this case is 6.



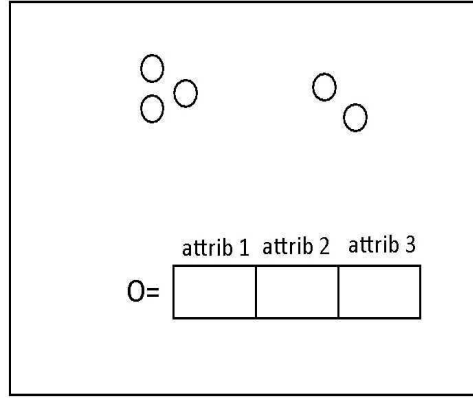


Figure 7.1: dataset of 5 points with 3 attributes.

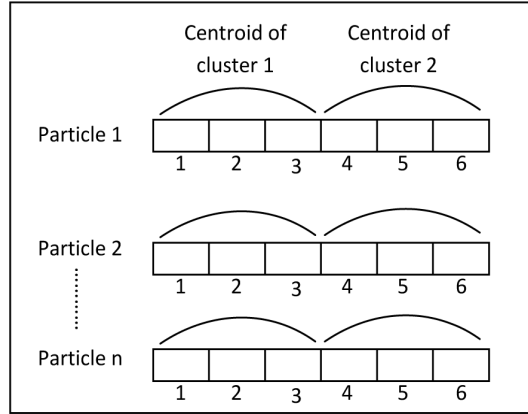


Figure 7.2: Encoding of particle's position when the number of clusters is  $k=2$

Based on the cluster centroids, each point in the dataset is assigned to the cluster with the nearest centroid. This is done by computing the Euclidean distance of the current point to each cluster's centroid, then assigning that point to the closest cluster. The pseudocode of the 'AssignPointsToClusters' procedure that carries out this task is given in Algorithm 18. This procedure allows to assign each data point into the corresponding cluster based on the centroid information. During the optimization process, the cluster centroids evolve over iterations by applying the dynamics of QPSO. Consequently, the

data points are reassigned to the new clusters. This process continues until a termination criterion is encountered.

---

**Algorithm 18** AssignPointsToClusters( ClustersCentroids, E)

---

```

1: Input: Centroids of the clusters, Dataset E
2: for i =1 to |E| do
3:   for k =1 to NumberOfClusters do
4:     DistanceToCluster(k) = Distance(E(i), ClustersCentroids(k)) {distance is the
       Euclidean distance }
5:   end for
6:   pointInCluster(i) = clusterNumber(min(DistanceToCluster))
7: end for
8: Output: pointInCluster

```

---

### 7.4.3 The objective Functions

The performance of a multi-objective clustering algorithm depends heavily on the selection of the clustering objectives [71]. In this work, compactness and connectivity have been chosen as the two complementary objectives to be optimized as they can measure the clustering quality from different aspects [50].

- **Cluster Compactness Measure**

This validity measure computes the overall deviation of a clustering by finding the overall sum of the distances between data points and their cluster centres. It is calculated according to the following equation given in [50]:

$$Comp(C) = \sum_{c_k \in C} \sum_{i \in c_k} \delta(i, \mu_k)$$

where  $C$  is the set of clusters,  $\mu_k$  is the centroid of cluster  $C_k$ , and  $\delta(.,.)$  is the selected distance function (Euclidean distance in our case). This objective should be minimized as it tends to keep the intra-cluster variation small [50].

- **Cluster Connectedness Measure**

This measure is based on the idea that the neighbouring data points should be placed in the same cluster. It is computed by the following equation [50]:

$$Conn(C) = \sum_{i=1}^N \left( \sum_{j=1}^L x_{i,nn_{ij}} \right)$$

$$Where \quad x_{r,s} = \begin{cases} \frac{1}{j}, & \text{if } \exists C_k : r \in C_k \wedge s \in C_k \\ 0, & \text{otherwise} \end{cases}$$

where  $nn_{ij}$  denotes the  $j^{th}$  nearest neighbour of point  $i$ ,  $N$  is the size of the dataset, and  $L$  is the number of neighbours of point  $i$ . This objective should also be minimized.

#### 7.4.4 Outline of MOQPSO for Clustering (MOQPSO-Clust)

Let  $E$  be an input dataset composed of points to be grouped into  $k$  clusters. Each point in  $E$  is defined by  $q$  attributes. Therefore, the dimension of the problem is  $D = k * q$ . Solving the clustering problem by the MOQPSO algorithm using the particle's position encoding described in section 7.4.2 requires first an initialization step where initial partitions are derived from  $k$  randomly generated centroids using the 'AssignPointsToClusters' procedure. The compactness and the connectivity values of each partition are computed and then an initial set of non-dominated solutions is created and set as the archive of the global best solutions or the current Pareto clustering solutions. Then, an iterative process is performed during which particles' positions are updated according to QPSO dynamics.

The hybrid selection method we developed is used as the leader selection scheme to find the global best guide for each solution. By using the ‘AssignPointsToClusters’ procedure, each data point is reassigned to the closest cluster according to the centroid information. Then, the obtained partitions are evaluated and the current Pareto set is updated. At the end of this process, the obtained Pareto optimal set along with the corresponding Pareto front are given as the output of the algorithm. Let us denote the set of non-dominated partitions or the archive of the global best clusterings in Pareto sense encountered at iteration  $t$  by  $GBA^t$ .  $S^t$  refers to the swarm of particles (cluster centroids) at iteration  $t$ . The proposed MOQPSO-Clust for data clustering can be described as follows:

---

**Algorithm 19** MOQPSO Clust

---

```

1: Input: MOP specification
2: N= population size
3: D= problem dimension
4:  $S^0$ = initialize positions of particles (cluster centroids) with uniformly distributed random numbers
5: pointInCluster = AssignPointsToClusters( $S^0$ , E) {assign data points to closest clusters}
6:  $sbest_i$ = initialize self best position of particle  $P_i$  for  $i=1..N$ 
7: T= maximum number of iterations
8:  $\vec{F}_i$ = evaluate Particle  $P_i$  for  $i=1..N$ 
9:  $GBA^0$ = initial set of non-dominated solutions (potential clustering solutions or partitions)
10:  $t = 1$ 
11:  $\beta^t = \beta_{max}$ 
12: repeat
13:   Compute mean best position using eq. (2.5) ;
14:   for (each particle  $P_i$ ) do
15:      $gbest = Select - leader(GBA^t, P_i)$ 
16:     for (each dimension  $j$ ) do
17:        $p_{ij}^t = \text{Compute local attractor using eq. (2.4)}$ 
18:        $u_i = rand(0, 1)$ 
19:       if  $rand(0, 1) > 0.5$  then
20:          $x_{ij}^{t+1} = p_{ij}^t + \beta^t \cdot |mbest_j^t - x_{ij}^t| \cdot \ln(1/u_{ij}^t)$  for  $j=1..D$ 
21:       else
22:          $x_{ij}^{t+1} = p_{ij}^t - \beta^t \cdot |mbest_j^t - x_{ij}^t| \cdot \ln(1/u_{ij}^t)$  for  $j=1..D$ 
23:       end if
24:     end for
25:      $pointInCluster = AssignPointsToClusters(S^t, E)$  {Reassign data points to new clusters}
26:     Evaluate particle  $P_i$ 
27:     Update self best position
28:   end for
29:    $GBA^{t+1} = Update - Archive(GBA^t, S^t)$ 
30:    $\beta^{t+1} = \beta^t - \frac{(\beta_{max} - \beta_{min})}{T}$  {Decrease  $\beta$  linearly}
31:    $t = t + 1$ 
32: until ( $t > T$ )
33: Output :  $GBA$ 

```

---

The algorithm has been adapted to comply with the clustering problems by introducing the objective functions as explained in section 7.4.3 and the ‘AssignPointsToClusters’ pro-

cedure described in Algorithm 18. This procedure is used at each iteration after updating the cluster centroids in order to reallocate the data points to the new clusters.

## 7.5 Time Complexity Analysis of MOQPSO-Clust for Data Clustering

We consider the following parameters that impact the size of the problem in this analysis:

- D: the problem dimension that is the number of decision variables
- M: the number of objective functions
- N: the population size
- L: the archive size and
- E: the number of data points in the dataset.

Data clustering is an NP-hard problem [5]. The number of partitions that can be obtained by grouping  $n$  data points into  $k$  clusters grows exponentially when the data size increases. The main operations of MOQPSO-Clust are the same as the main operations of MOQPSO for unconstrained problems described in section 3.5. These operations are:

1. Update of particles' positions with time complexity  $O(N^2)$
2. Evaluation of positions with time complexity  $O(NM)$
3. Leader selection with time complexity  $O(MN^2 \log N)$
4. Update of the archive or Pareto front with time complexity  $O(MN^2)$

The main difference between the basic MOQPSO proposed to solve unconstrained problems in chapter 3 and the MOQPSO-clust is the ‘Assign points-to-cluster’ procedure. From Algorithm 18, the distance of each data point to each centroid is computed in order to determine the closest centroid. Therefore, the time complexity of this procedure is  $O(Nkq|E|)$  where  $k$  denotes the number of clusters and  $q$  denotes the dimension of a data point or the number of its attributes, which can be expressed as well as  $O(ND|E|)$  knowing that the dimension of the problem  $D$  is equal to  $k * q$ . The time complexity of the other parts of MOQPSO-Clust remains the same as that of MOQPSO for unconstrained problems, which is  $O(MN^2 \log N)$  as described in section 3.5. Hence, the overall time complexity of MOQPSO-Clust is  $O(MN^2 \log N + ND|E|)$ .

Since MOQPSO-Clust scales like  $O(MN^2 \log N + ND|E|)$ , it is not suitable for very big datasets. However, clustering is important for image segmentation, data mining, speech recognitions, visualization of scientific data, and many others that do not require large datasets but require a trade off of solutions to be presented to the user from which the user can choose the appropriate solution. MOQPSO-Clust is suitable for this purpose.

Further work will be required to improve the abilities of the proposed MOQPSO-Clust algorithm to handle clustering of big data. According to the computational complexity of MOQPSO-Clust, it is clear that a prohibitively high computational cost will be induced if applied to big data due to the data size, the multi-objective nature of the problem, the high dimension of data and the expensive process of evaluating the quality of solutions which is problem specific.

## 7.6 Experiments

In this section, we describe the experimental setup and the results of the several experiments of MOQPSO-Clust performed for comparison with other clustering techniques.

### Data Sets

The proposed algorithm has been applied to both synthetic and real world datasets. The synthetic datasets are generated by the Gaussian cluster generator described in [49]. Real datasets are taken from the UCI machine learning databases repository [16]. The detailed characteristics of the employed datasets are illustrated in Tables 7.1 and 7.2 where  $K$  denotes the number of clusters,  $Dim$  is the dimension of data point,  $size$  is the dataset size, and  $n_i$  is the number of the points in each cluster [88].

The main goal of this chapter is to investigate the applicability of MOQPSO to the clustering problem. That is why these datasets have been chosen in our study since they are well known datasets in the clustering field. Thus, experimental results from a variety of other clustering algorithms are commonly available, which facilitates the comparison with the proposed algorithm. In addition, the ground truth solutions of these datasets are available, which simplifies the estimation of the performance of the algorithm.

Tabel 7.1: Characteristics of the used synthetic datasets

Dataset Name	K	Dim	Size	$n_i$
2d4c	4	2	1123	369,471,53,230
2d10c	10	2	520	67, 15, 19, 53, 83, 64, 65, 68, 68, 18
10d10c	10	10	436	18, 83, 57, 26, 67, 50, 12, 72, 39, 12

Tabel 7.2: Characteristics of the used real datasets

<b>Dataset Name</b>	<b>K</b>	<b>Dim</b>	<b>Size</b>	$n_i$
Ruspini	4	2	75	20, 23, 17, 15
Iris	3	4	150	50,50,50
Wisconsin	2	9	699	458, 241

## Evaluations

Two cluster validity measures have been used in this study in order to assess the quality of the final partitioning result produced by the proposed MOQPSO-Clust algorithm for solving clustering problems and to conduct a comparative study with the classical method (K-means). Generally speaking, there are two types of cluster validity methods, external and internal validations. The external criteria relies on prior knowledge or external information about the data to perform the evaluation. They are used when ground truth data are available. On the other hand, the evaluation process with the internal criteria is based on information that is only inherent to the data [47].

There are many cluster validity indexes that have been proposed in the literature [47][91]. In our study, we will use one external validation measure called F-measure [91] and one internal validation measure called Silhouette index [91] for measuring the quality of the partitions produced by the proposed MOQPSO-Clust.

- **F-measure**

It is one of the commonly used validity measures in the specialized literature. It measures the degree of similarity of an obtained clustering to each ground truth class of the given dataset. Assume that  $GC = (GC_1, GC_2, \dots, GC_K)$  denotes the ground truth classes of the dataset and  $C = (C_1, C_2, \dots, C_K)$  denotes the obtained clustering result. Then the F-measure of cluster  $C_i$  and class  $GC_j$  is given by the following equation [102]:



$$F(C_i, GC_j) = \frac{2|C_i \cap GC_j|}{|C_i| + |GC_j|}$$

the overall F-measure of a clustering  $C$  with respect to  $GC$  is given by [102]:

$$F(C, GC) = \sum_j \frac{|GC_j|}{m} \max_i F(C_i, GC_j)$$

where  $m$  is the dataset size. The F-measure values are within the range  $[0,1]$ . The larger the F-measure values, the higher the quality of the clustering [102].

- **Silhouette index**

This index measures how well each point lies within its cluster. It actually measures both cluster cohesion (intra-cluster) and separation (inter-cluster). Let  $a$  = the average distance of a data point  $i$  to all remaining points in its cluster and let  $b$  = the minimum average distance of point  $i$  to the points in the other clusters. Then the silhouette of point  $i$   $s(i)$  is given by [91]:

$$s(i) = \frac{(b_i) - (a_i)}{\max\{(a_i), (b_i)\}}$$

The average  $s(i)$  for the entire dataset measures the goodness of the clustering result. The silhouette index values are within the interval  $[-1, 1]$ . The closer the value of this measure to 1, the higher the quality of the clustering [49][71].

### 7.6.1 Experimental Set Up

Preliminary experiments have been conducted to set the algorithm's parameter, namely the contraction expansion parameter  $\beta$  which varies within the range  $[1.2 - 0.5]$  as it shows good results in terms of convergence and diversity within this interval. In order to set the remaining problem parameters properly (number of iterations and number of particles), several experiments have been conducted. We started the experiments by running the algorithm with a small number of particles (30, 50, 80) and a small number of iterations

(50, 80, 100) and plotting the obtained Pareto fronts at the end of each run. We continue repeating the run process by increasing the number of particles (100, 120, 150) and the number of iterations (120, 150, 200) until we reach a good convergence and diversity of the Pareto fronts for all the datasets. At the end, we got good quality solutions with the following setting: the number of particles is set to 150 and the number of iterations is set to 200 for all the datasets even good results have been obtained with less values as in the case of 2d4c and Ruspini.

### 7.6.2 Experimental Results and Discussions

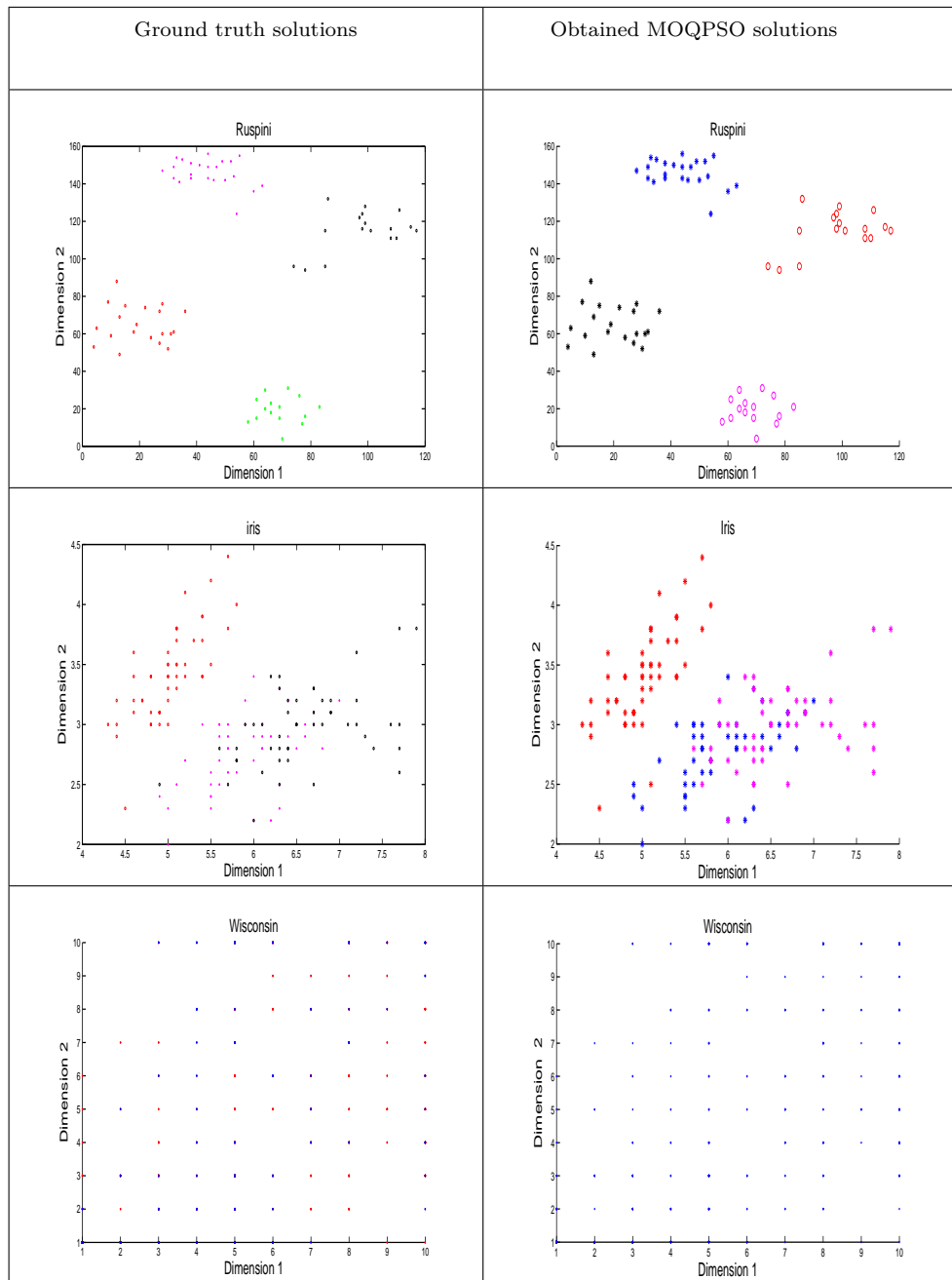
At first, we have tested the ability of the proposed algorithm to solve the clustering problem on the used datasets. Tables 7.3 and 7.4 show the figures of the ground truth partitions and the obtained partitions using MOQPSO for the real and the synthetic data sets respectively. It can be seen from these figures that MOQPSO was able to find out good partitions especially in the case of 2d4c, 10d10c and Ruspini.

Furthermore, the obtained results reveal that the algorithm is effective and flexible in providing a set of diverse partitioning solutions. For example, in the case of 2d4c dataset, the proposed algorithm provides nine trade-off clustering solutions for the end users so that they can choose the final solution according to their preferences. Figures 7.3 and 7.4 show these nine Pareto clustering solutions. In all Pareto solutions, MOQPSO was able to find the core partitions. The difference between these partitions lies in the boundaries of the clusters. This is particularly interesting in the case of overlapping clusters.

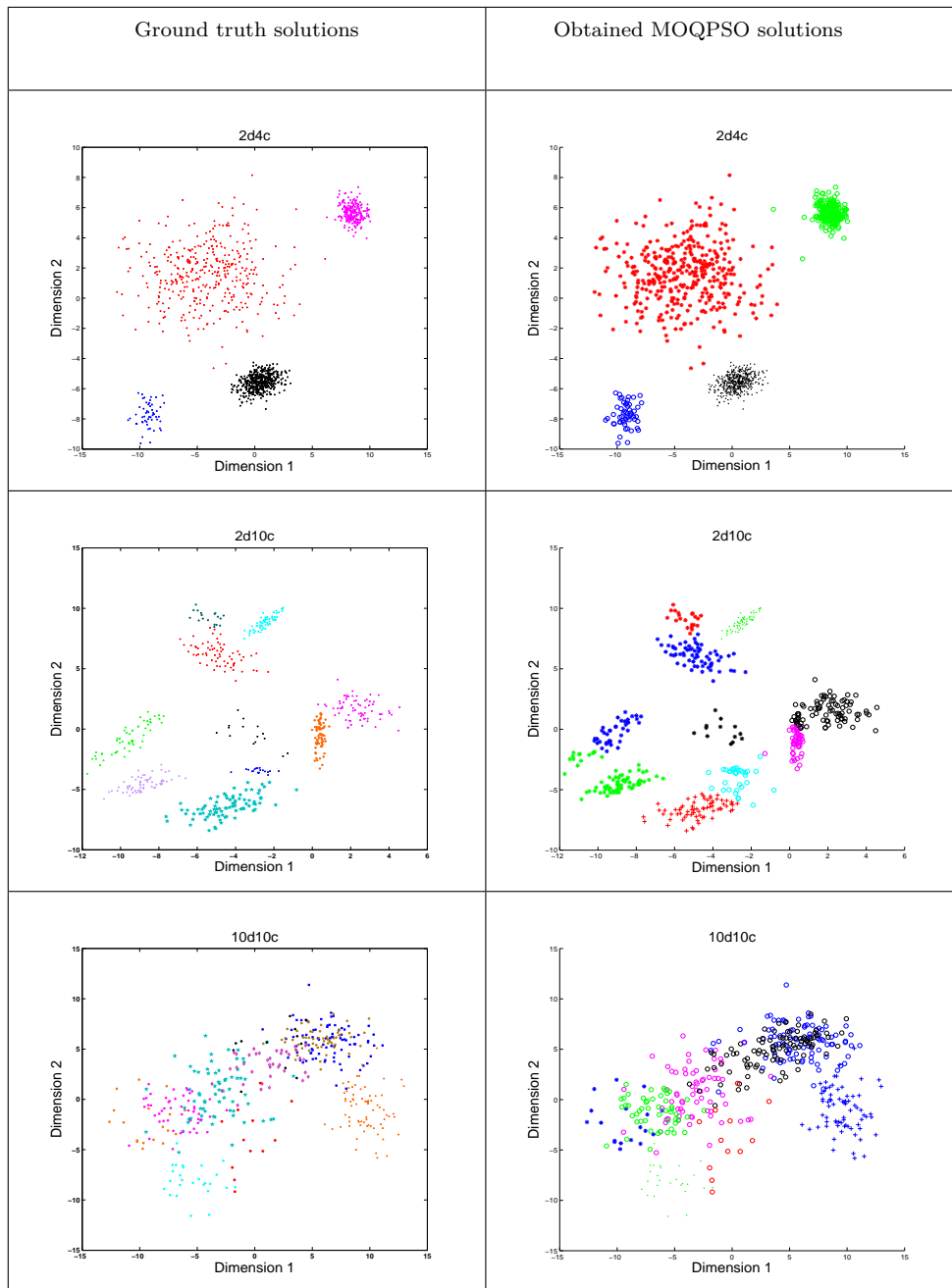
For a quantitative assessment, the algorithm has been run 30 times for each dataset and the best solutions in terms of F-measure and silhouette index have been recorded. K-means algorithm has been run the same number of times on the same datasets. Then,

statistics have been derived using the obtained partitions of both algorithms. Tables 7.5 and 7.6 show the values of F-measure and silhouette index respectively using MOQPSO-Clust and K-means on the employed datasets. As can be seen from the F-measure results, our algorithm outperformed K-means in all cases in terms of best values except for 2d10c dataset where K-means performed slightly better. However, MOQPSO achieved best results at the average for all data sets. Furthermore, our algorithm exhibits more stability than K-means as shown in the difference between mean values and median ones as shown in Table 7.5.

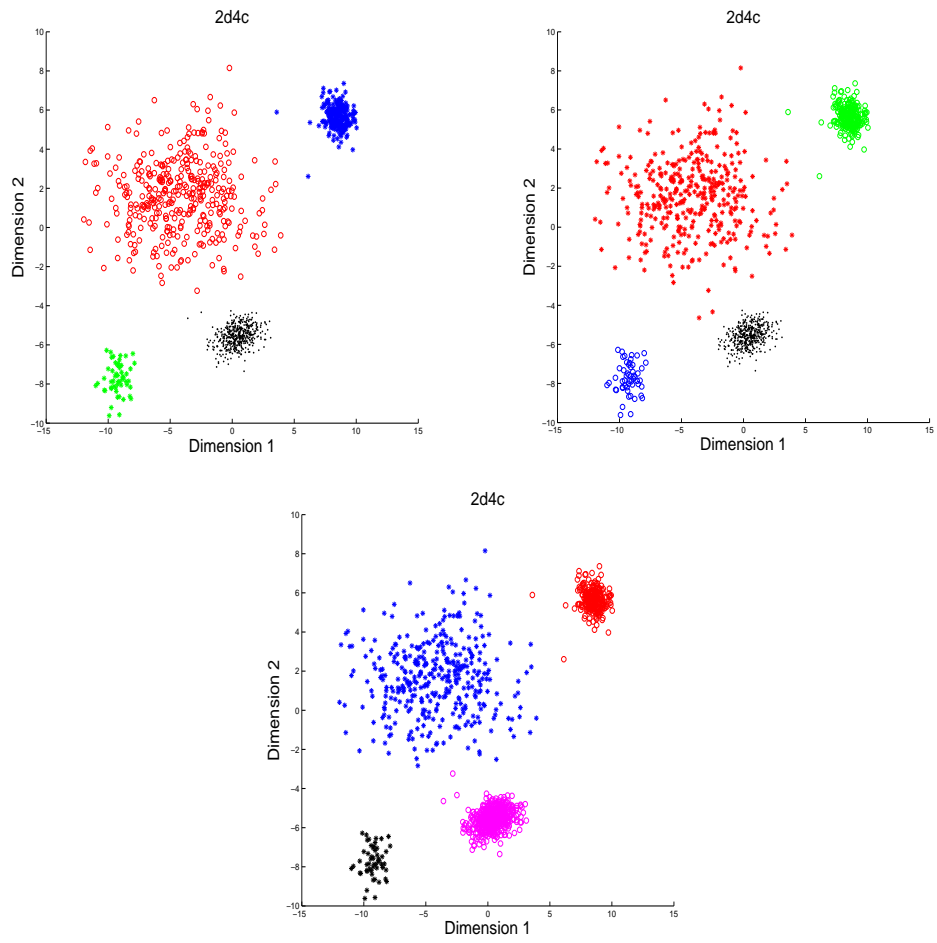
As for silhouette results, we can see that competitive results have been achieved. Our algorithm performed better at the average except for 2d10c and Wisconsin cases. We can see as well that both algorithms were able to find the optimal solution in case of 10d10c and Ruspini datasets. Moreover, MOQPSO-Clust has been compared with five other clustering algorithms from the literature, namely FCM (Fuzzy C-Means), PCA (Possibilistic Clustering Algorithm), UPFC (Unsupervised Possibilistic Fuzzy Clustering), IQEAC (Improved Quantum Evolutionary Algorithm for data Clustering) and GA (Genetic Algorithm). These algorithms have been chosen because they are recently proposed methods for clustering and they demonstrate the state-of-the-art performance. In addition, some of these algorithms are evolutionary algorithms such as IQEAC and GA. Table 7.7 shows the results in terms of median and interquartile of F-measure using the proposed MOQPSO-Clust algorithm and the five competing algorithms with settings as reported in [87]. Best results have been obtained by our algorithm on 2d4c, 10d10c and Ruspini datasets. It also achieved better results than all other algorithms except for IQEAC on 2d10c. Intermediate results have been achieved with Iris and Wisconsin datasets.



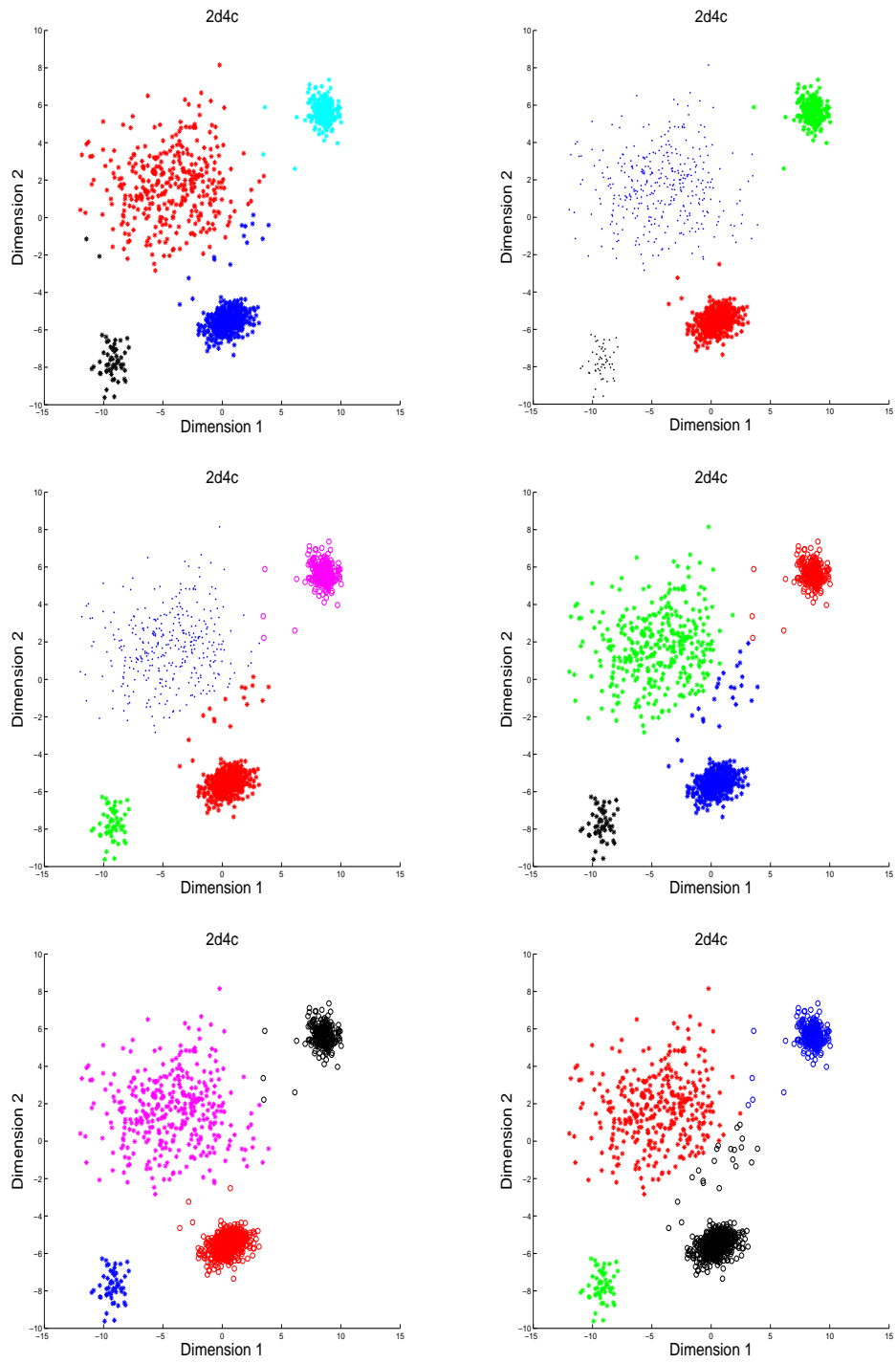
Tabel 7.3: Obtained clustering results vs truth clustering results for real data sets



Tabel 7.4: Obtained clustering results vs truth clustering results for synthetic data sets



Figur 7.3: Pareto solutions representing the trade-off partitions for 2d4c dataset (the first three solutions).



Figur 7.4: Pareto solutions representing the trade-off partitions for 2d4c dataset (the remaining six solutions).

	MOQPSO			K-means		
Data sets	Best	Mean (Std.)	Median	Best	Mean (Std.)	Median
2d4c	0.9982	0.9982 (0.00)	0.9982	0.9730	0.9730(0.00)	0.9730
2d10c	0.9031	0.9031(0.00)	0.9031	0.9190	0.7907(0.0595)	0.7757
10d10c	1.00	0.9965(0.0110)	1.00	1.00	0.8231(0.1231)	0.8406
Ruspini	1.00	1.00(0.00)	1.00	1.00	0.8509(0.1058)	0.8171
Iris	0.9597	0.8605(0.0531)	0.8396	0.8918	0.8323(0.0958)	0.8918
Wisconsin	0.9243	0.9242( $1.399 * 10^{-4}$ )	0.9241	0.8733	0.8733(0.00)	0.8733

Tabel 7.5: F-measure results using MOQPSO-Clust and K-means on synthetic and real data sets.

	MOQPSO			K-means		
Data sets	Best	Mean (Std.)	Median	Best	Mean (Std.)	Median
2d4c	0.8615	0.8610(0.0010)	0.8615	0.8628	0.8628( $1.170 * 10^{-16}$ )	0.8628
2d10c	0.6973	0.6973(0.00)	0.6973	0.7216	0.6842(0.0312)	0.6889
10d10c	0.8194	0.8165(0.0093)	0.8194	0.8194	0.6098(0.1487)	0.6403
Ruspini	0.9086	0.9086(0.00)	0.9086	0.9086	0.7549(0.1073)	0.7024
Iris	0.8462	0.8462( $1.1703 * 10^{-16}$ )	0.8462	0.7355	0.7121(0.0377)	0.7355
Wisconsin	0.7526	0.7508(0.0015)	0.7499	0.7550	0.7550(0.00)	0.7550

Tabel 7.6: Silhouette measure results using MOQPSO-Clust and K-means on synthetic and real data sets.

Data sets	MOQPSO	FCM	PCA	UPFC	IQEAC	GA
2d4c	0.9982(0.00)	0.9392(0.00)	0.7428(0.1877)	0.8134(0.0444)	0.9784(0.00)	0.9730(0.00)
2d10c	0.9031(0.00)	0.8861(0.0681)	0.7320(0.1039)	0.7671(0.0804)	0.9582(0.0066)	0.9027(0.0506)
10d10c	1.00(0.00)	0.9254(0.00)	0.6666(0.1122)	0.7747(0.0833)	1.00(0.00)	0.9344(0.0353)
Iris	0.8396(0.0153)	0.8923(0.00)	0.7233(0.2367)	0.9061(0.00)	0.8988(0.00)	0.8923(0.0005)
Wisconsin	0.9241(0.0002)	0.9558(0.00)	0.6812(0.3685)	0.9588(0.00)	0.9677(0.00)	0.9662(0.0014)
Ruspini	1.00(0.00)	-	-	-	-	-

Tabel 7.7: Comparison with other algorithms based on Median (Interquartile). The results in this table are quoted from [87].



## 7.7 Summary

In this chapter, we have presented the application of the proposed MOQPSO framework in solving partitional clustering problem. To our knowledge, it is the first Pareto-based MOQPSO specifically designed for this purpose. The experimental results demonstrate the ability of MOQPSO to handle the clustering problem. The main feature of this approach is its ability to provide the end users with multiple optimal clustering options from which a partition can be chosen according to their specific needs. Our algorithm has been found to perform successfully on both synthetic and real data sets. The proposed MOQPSO-based approach outperformed k-means in most cases and shows competitive results compared to other algorithms. The computational scaling of the current version of MOQPSO-Clust, which is  $O(MN^2 \log N + ND|E|)$ , is not suitable for big data applications and this is an interesting area where future work will be needed.

### Conclusions and Future Work

---

In this chapter, we will provide a summary of the work and the contributions described in this thesis followed by an outline of some possible future research directions that could be drawn from the present work.

#### 8.1 Summary of the Work

Most real-world decision problems have multiple objectives that have to be optimized simultaneously. Many of these problems are subject to some constraints. In this thesis, we have begun with a review of the most widely used multi-objective evolutionary algorithms for solving multi-objective optimization problems. We then developed a new framework by extending Quantum behaved Particle Swarm Optimization(QPSO) to handle unconstrained multi-objective problems.

In our framework, which we called (MOQPSO), we address the way global best solutions are recorded within an archive and used to compute the local attractor point of each particle. For this purpose, a two level selection strategy that uses sigma values and crowding distance information has been defined in order to select the suitable guide for

each particle. The rationale has been to help convergence of each particle using sigma values while favoring less crowded regions in the objective space to attain a uniformly spread out Pareto front. The time complexity of MOQPSO is  $O(MN^2 \log N)$  which shows a polynomial scaling with respect to the parameters of the problem. Besides, it is only by a log factor larger than the state-of-the-art MOEAs such as NSGAI, PEAS and SPEA, which is considered a little overhead. The proposed approach has been assessed on benchmark test problems from convergence and diversity points of view and compared against some state-of-the-art multi-objective algorithms showing competitive results.

After the encouraging results of MOQPSO for solving unconstrained MOPs, we developed CMOQPS, which extended the MOQPSO framework to handle constrained multi-objective problems. Two strategies to handle constraints are investigated. The first one is the death penalty strategy which discards infeasible solutions that are generated through iterations forcing the search process to explore only the feasible region. The second approach which we proposed keeps the infeasible solutions when computing the local attractors of particles and adopts a policy that achieves a balance between searching in infeasible and feasible regions. The aim of this latter approach is to incorporate the infeasible solutions during the evolutionary process hoping to find good quality feasible solutions. Several benchmark test problems have been used for testing and validation. Experimental results show the ability of QPSO to handle constraints effectively in multi-objective context. The first constraint handling strategy has been found to be the best in all cases in terms of convergence. It also has achieved better results in terms of diversity for most of the test problems. However, it requires an additional computational effort of  $O(RND)$  when compared with the second constraint handling strategy. On the other hand, the second constraint handling strategy scales as the MOQPSO for unconstrained problems. In the future work section we give a suggestion of how the second constraint handling

strategy could be improved.

Further to this work, a thorough investigation of the potential of MOQPSO under several leader selection strategies (the proposed hybrid selection strategy, the crowding distance based selection strategy, the sigma based selection strategy, and the random selection strategy) and several archiving methods (unbounded archive size, clustering, crowding, maximin, and the proposed redundancy removal method) on unconstrained and constrained test problems have been provided. In addition, a comparative study of MOQPSO with MOPSO is performed.

The results obtained from the comparison of the leader selection strategies showed that the sigma method presented better convergence metric values but poor diversity and the crowding selection method obtained better diversity values yet poor convergence. However, the proposed hybrid selection method succeeded in maintaining a balance in obtaining good convergence and good diversity values for all the test functions.

As for the archiving methods, we proposed the redundancy removal archiving method, a new simple yet effective mechanism to handle the growth of the archive size. The aim of this technique has been to take advantage from both the bounded and the unbounded archiving methods. That is, getting the advantage of the unbounded archive method in reaching good convergence to the Pareto front and maintaining good diversity. At the same time, it takes the advantage of the bounded archiving methods in reducing the computational time of the algorithm and providing a smaller number of non-dominated solutions to the end users. Results of the comparative study have shown that the best convergence results have been obtained with the unbounded archive size and the maximin methods. The best diversity results have been achieved with the unbounded archive size

and the clustering methods for unconstrained test problems and the clustering and the crowding methods for constrained test problems. The proposed redundancy removal method has shown competitive results compared to the unbounded archive method with a significant reduction in CPU time for both constrained and unconstrained test problems. The redundancy removal method and the bounded archiving strategies scale linearly with the number of objective functions. However, better time complexity is achieved by the crowding based archiving strategy as it scales quasi-linearly with the archive size while the other archiving methods have a quadratic scaling with the archive size.

The comparison results of MOQPSO against MOPSO revealed that none of the two algorithms has been found to be the best in all cases. QPSO was highly competitive when compared with PSO in terms of convergence and diversity. QPSO was able to obtain quality Pareto fronts for some of the test problems. It even outperforms PSO in ZDT2 function as PSO failed to solve this function due to the premature convergence problem.

After testing the performance of the proposed algorithm and demonstrating its effectiveness in solving unconstrained and constrained test problems and under different selection and archiving strategies, we intended to investigate the potential of MOQPSO in solving real-world application domains. We applied our framework for solving the cluster analysis problem which we called MOQPSO-Clust. We cast clustering as a Pareto based multi-objective optimization problem which is handled using a quantum behaved particle swarm optimization algorithm. The search process is performed over the space of cluster centroids with the aim to find partitions that optimize two objectives simultaneously. The proposed hybrid selection method is used as the leader selection strategy to select the global best leader for each particle. The main objectives of this study are to demonstrate the ability of MOQPSO to handle the clustering problem and the ability

of MOQPSO-Clust to obtain meaningful trade-off partitions. MOQPSO-Clust has been tested using both synthetic and real datasets and compared to the state-of-the-art methods showing competitive results. As MOQPSO-Clust scales like  $O(MN^2 \log N + ND|E|)$ , it is not suitable for big data applications. Future work will be required to improve the abilities of the proposed MOQPSO-Clust algorithm to handle clustering of big data.

Generally, MOQPSO algorithm performs well when solving continuous and discontinuous Pareto fronts in terms of convergence and diversity even without the need of incorporating any diversity preserving mechanism. This is due to the dynamics of the QPSO algorithm that allows to maintain a good balance between exploration and exploitation in the search space. The merits of the proposed MOQPSO algorithm can be summarized in the following points:

- Its simplicity and the few number of tunable parameters.
- The leader selection strategy which helps to achieve a good balance between diversity and convergence of obtained Pareto fronts compared to other strategies.
- The redundancy removal strategy that does not impose any limit on the archive size which may impact the quality of the obtained fronts from diversity and convergence points of view. It also ensures a saving in run time compared to the unbounded archive method while achieving comparable quality of the obtained fronts.
- Ability to deal efficiently with disconnected fronts as seen in the case of ZDT3 test function where MOQPSO gave significantly better results in comparison with the state-of-the-art MOEAs, namely NSGAI, PAEAS and SPEA. This is due to the efficient global search capability of QPSO that is able to jump through the different regions of the search space, and the effective influence of the proposed hybrid selection strategy that helps to achieve good quality solutions from convergence and

diversity points of view.

We have seen in extensive experiments that our proposed approach was able to achieve better results compared to state-of-the-art methods with a very minor extra burden ( $\log N$ ) on computational complexity.

On the other hand, the limitations of the proposed MOQPSO can be summarized in the following points:

- MOQPSO faces some difficulties in maintaining diversity when the number of solutions decreases near the Pareto optimal front. This loss of diversity prevents the algorithm from converging properly to the optimal fronts. This case has been seen in the ZDT6 test function where the number of solutions decreases near the Pareto optimal front. Introducing a diversity preserving operator (mutation) will perhaps enhance the search abilities of the algorithm in this case.
- Time complexity of MOQPSO does not allow application to big data problems. It is not obvious how this approach could be scaled up, this will require further research.
- The algorithm has been only tested on two objective optimization problems. Dealing with many objectives is a large area of research as for example the PhD thesis of Praditwong [84] deals exclusively with the many objective optimization problems. It will be interesting in the future work to investigate to what extend the proposed approach can solve problems with many objectives and at which number of objectives it may break down as it known that many objective problems require special treatment.

## 8.2 Future Work

As mentioned earlier in section 3.2.1, the strategy we follow in our work is to keep only one solution as the self best point (*sbest*) for each particle and use the GBA archive to store the non-dominated solutions obtained by all particles in the swarm from which the global best leader is selected. There are different approaches in the literature that are based on saving all the non-dominated solutions visited by each particle in an archive and then select the self best particle among them [1] [99] [17]. The results show that using two external archives for global best and self best positions would improve the performance and effectiveness of the algorithm. We would like to explore the benefits of using this feature in our proposed MOQPSO algorithm by using one archive which we can call Self Best Archive (SBA) to store the non-dominated solutions visited by each particle in the swarm together with the GBA archive.

When extending QPSO to handle constraints, we found that the proposed constraint handling strategy (the second strategy) which incorporates the infeasible solutions during the evolutionary process was not able to get the benefits from the infeasible solutions in obtaining good feasible ones. We believe that this might be related to the constraint handling scheme we adopted, which is based mainly on the number of constraint violations when evaluating the infeasible solutions. We might need to improve the constraint handling mechanism by introducing another criterion when dealing with infeasible solutions which is based on calculating the distance of infeasible solutions to the boundary of the feasible region.

In chapter 7, we have presented a multi-objective quantum behaved particle swarm optimization algorithm for solving partitional clustering problem. The main feature of



this approach is its ability to provide the end users with multiple optimal clustering options from which a partition can be chosen according to their specific needs. Extension to the current approach should be studied to include automatic detection of the number of clusters. Additionally, the approach should be tested on more complex datasets that exhibit higher volume and dimension of data points as well as complex distributions with non-linearly separable clusters.

Furthermore, it would be interesting to study the effect of introducing the mutation operator to the proposed MOQPSO algorithm and demonstrate how this will improve the diversity of the obtained solutions.

Another area of future work is to study the behavior of the proposed algorithm when solving many-objective optimization problems (more than two or three objectives) as they are difficult to solve and there is a little work done in this area. Recent studies have shown the limitations of MOP methods when many objectives need to be considered. Indeed, dealing with many objectives raises several challenges that should be properly addressed. This issue is at the heart of recent and ongoing research activities. One of these challenges is that many objectives imply high dimensional objective spaces. Therefore, a good approximation of the Pareto-front requires in this case a huge number of points which increases exponentially with the number of objectives resulting in turn in a high computational cost and a difficulty of visualization [94]. Furthermore, other challenges are related to the search ability and the requirements of each method. For the weighted sum approach, determining the appropriate weights becomes even more difficult when the number of objectives increases especially when no further information about the problem is available. Furthermore, it has been shown to deal poorly with non-convex search spaces - a situation which is worsened with the increase in the number of objectives. In VEGA,

a subpopulation is assigned to each objective, making selection in each sub-population based only on this objective without considering the other objectives. This leads in turn to diversity reduction as it favors solutions performing well in one dimension and ignores solutions that perform reasonably well in all dimensions [44]. This in turn results in poor scalability in many objectives context. Furthermore, it has the same issue as the weighted sum approach regarding non-convexity of search spaces. This will add an additional difficulty when dealing with many objectives.

For MOEAs like NSGAI, SPEA2, and PAES, the problem of diversity loss limits their scalability to many objectives. This issue can be explained by the fact that they are density based and favor Pareto dominance selection while focusing on remote and boundary solutions [94].

For the case of swarm based approaches, as seen before maintaining diversity relies heavily on the used leader selection strategy and the search ability governed by the PSO model equations. For example, the MOPSO approach proposed by Coello [20] used the adaptive grid scheme to select the suitable guide for each particle. This technique does not scale well when dealing with many-objective problems because the update of the adaptive grid becomes more difficult and time consuming.

Although dealing with many-objective problems is a non-trivial task, it can be considered as a very interesting line of research in the future.



---

## Bibliography

---

- [1] Mohammad Ali Abido. Two-level of nondominated solutions approach to multiobjective particle swarm optimization. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, pages 726–733. ACM, 2007. 2 citations in sections 2.3.4 and 8.2.
- [2] Ajith Abraham, Swagatam Das, and Sandip Roy. Swarm intelligence algorithms for data clustering. In *Soft Computing for Knowledge Discovery and Data Mining*, pages 279–313. Springer, 2008. 2 citations in sections 7.1 and 7.3.
- [3] Ajith Abraham and Lakhmi Jain. *Evolutionary multiobjective optimization*. Springer, 2005. 3 citations in sections 1.2, 2.1.3, and 2.1.3.
- [4] Hamid Ali, Waseem Shahzad, and Farrukh Aslam Khan. Energy-efficient clustering in mobile ad-hoc networks using multi-objective particle swarm optimization. *Applied Soft Computing*, 12(7):1913–1928, 2012. One citation in section 7.3.1.
- [5] Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. Np-hardness of euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248, 2009. One citation in section 7.5.
- [6] Julio E Alvarez-Benitez, Richard M Everson, and Jonathan E Fieldsend. A mopso algorithm based exclusively on pareto dominance concepts. In *Evolutionary Multi-Criterion Optimization*, pages 459–473. Springer, 2005. One citation in section 2.3.4.
- [7] Pierre Baldi et al. *Bioinformatics: the machine learning approach*. The MIT Press, 2001. One citation in section 7.1.

- [8] Richard Balling. The maximin fitness function; multi-objective city and regional planning. In *Evolutionary Multi-Criterion Optimization*, pages 1–15. Springer, 2003. One citation in section 2.3.4.
- [9] Alexandre M Baltar and Darrell G Fontane. A generalized multiobjective particle swarm optimization solver for spreadsheet models: application to water quality. *Proceedings of the Twenty Sixth Annual American Geophysical Union Hydrology Days*, pages 20–22, 2006. One citation in section 1.2.
- [10] Sanghamitra Bandyopadhyay, Ujjwal Maulik, and Anirban Mukhopadhyay. Multiobjective genetic clustering for pixel classification in remote sensing imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 45(5):1506–1511, 2007. One citation in section 7.3.1.
- [11] Alec Banks, Jonathan Vincent, and Chukwudi Anyakoha. A review of particle swarm optimization. part i: background and development. *Natural Computing*, 6(4):467–484, 2007. One citation in section 2.2.1.
- [12] De-hai Bao. An improved algorithm of quantum-behaved particle swarm optimization and its application in time-table problem of universities. *IERI Procedia*, 2:534–537, 2012. One citation in section 1.2.
- [13] Thomas Bartz-Beielstein, Philipp Limbourg, Jörn Mehnen, Karlheinz Schmitt, Konstantinos E Parsopoulos, and Michael N Vrahatis. Particle swarm optimizers for pareto optimization with enhanced archiving techniques. In *The 2003 Congress on Evolutionary Computation, 2003. CEC’03*, volume 3, pages 1780–1787. IEEE, 2003. One citation in section 2.3.4.
- [14] James C Bezdek. *Pattern recognition with fuzzy objective function algorithms*. Kluwer Academic Publishers, 1981. One citation in section 7.3.1.
- [15] To Thanh Binh and Ulrich Korn. MOBES: A multiobjective evolution strategy for constrained optimization problems. In *The Third International Conference on Genetic Algorithms (Mendel 97)*, pages 176–182. Citeseer, 1997. 4 citations in sections 2.1.2, 2.1.3, 2.1.5, and 4.4.
- [16] Catherine Blake and Christopher J Merz. UCI, repository of machine learning databases. 1998. One citation in section 7.6.

- [17] Jürgen Branke and Sanaz Mostaghim. About selecting the personal best in multi-objective particle swarm optimization. In *Parallel Problem Solving from Nature-PPSN IX*, pages 523–532. Springer, 2006. 2 citations in sections 2.3.4 and 8.2.
- [18] Claudio Carpineto and Giovanni Romano. A lattice conceptual clustering system and its application to browsing retrieval. *Machine Learning*, 24(2):95–122, 1996. One citation in section 7.1.
- [19] John W Chinneck. Practical optimization: a gentle introduction. *Systems and Computer Engineering, Carleton University, Ottawa*. <http://www.sce.carleton.ca/faculty/chinneck/po.html>, 2006. One citation in section 2.1.1.
- [20] Carlos A Coello and Maximino Salazar Lechuga. MOPSO: A proposal for multiple objective particle swarm optimization. In *Proceedings of the 2002 Congress on Evolutionary Computation, 2002. CEC’02*, volume 2, pages 1051–1056. IEEE, 2002. 3 citations in sections 2.3.4, 2.4, and 8.2.
- [21] Carlos A Coello Coello, Clarisse Dhaenens, Laetitia Jourdan, et al. *Advances in Multi-Objective Nature Inspired Computing*, volume 272. Springer, 2010. One citation in section 1.1.
- [22] Carlos A Coello Coello, Gregorio Toscano Pulido, and M Salazar Lechuga. Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):256–279, 2004. 3 citations in sections 2.1.3, 4.1, and 5.1.2.
- [23] Carlos A Coello Coello, David A Van Veldhuizen, and Gary B Lamont. *Evolutionary algorithms for solving multi-objective problems*, volume 242. Springer, 2002. One citation in section 2.3.
- [24] Carlos A Coello Coello and Col San Pedro Zacatenco. 20 years of evolutionary multi-objective optimization: what has been done and what remains to be done. *Computational Intelligence: Principles and Practice*, pages 73–88, 2006. 3 citations in sections 2.3.2, 2.3.3, and 2.3.4.

- [25] Carlos Coello Coello, Gary B Lamont, and David A Van Veldhuizen. *Evolutionary algorithms for solving multi-objective problems*. Springer, 2007. One citation in section 2.1.3.
- [26] Carlos A Coello Coello. Constraint-handling using an evolutionary multiobjective optimization technique. *Civil Engineering Systems*, 17(4):319–346, 2000. One citation in section 4.1.
- [27] Carlos Artemio Coello Coello. An empirical study of evolutionary techniques for multiobjective optimization in engineering design. *PhD Thesis, Department of Computer Science, Tulane University, New Orleans, LA*, 1996. One citation in section 2.3.4.
- [28] WJ Conover. Practical nonparametric statistics, 1999. One citation in section 5.1.1.
- [29] David W Corne, Nick R Jerram, Joshua D Knowles, Martin J Oates, et al. PESA-II: Region-based selection in evolutionary multiobjective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)*. Citeseer, 2001. One citation in section 7.3.1.
- [30] George B Dantzig and Mukund N Thapa. *Linear Programming 2: Theory and Extensions*, volume 2. Springer Science & Business Media, 2003. One citation in section 2.1.2.
- [31] Kalyanmoy Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Chichester : Wiley, 2001. 21 citations in sections (document), 1.1, 1.2, 2.1.1, 2.1.2, 2.1.3, 2.2, 2.1.3, 2.3, 2.4, 2.1.3, 2.5, 2.1.3, 2.3.1, 2.3.2, 2.3.3, 2.3.4, 3.3, 3.6.2, 4.1, and 4.4.
- [32] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, pages 182–197, 2002. 11 citations in sections 2.3.4, 3.2.1, 3.3, 3.3, 3.6.1, 3.6.2, 3.6.4, 3.6.4, 4.1, 6.1, and 6.2.
- [33] Kalyanmoy Deb, Amrit Pratap, and T Meyarivan. Constrained test problems for multi-objective evolutionary optimization. In *Evolutionary Multi-Criterion Optimization*, pages 284–298. Springer, 2001. One citation in section 4.4.

- [34] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006. 3 citations in sections 4.4.3, 5.1.1, and 5.1.2.
- [35] Gerry Dozier, Shaun McCullough, Abdollah Homaifar, Eddie Tunstel, and Loretta Moore. Multiobjective evolutionary path planning via fuzzy tournament selection. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 684–689. IEEE, 1998. One citation in section 2.3.2.
- [36] Frank Drewes. Lecture notes on computational complexity. 2013. One citation in section 2.1.4.
- [37] Wei Fang, Jun Sun, Yanrui Ding, Xiaojun Wu, Wenbo Xu, et al. A review of quantum-behaved particle swarm optimization. *IETE Technical Review*, 27(4):336–348, 2010. 5 citations in sections 1.2, 2.2.2, 2.2.2, 2.2.2, and 3.2.3.
- [38] Jonathan E Fieldsend and Sameer Singh. A multi-objective algorithm based upon particle swarm optimisation, an efficient data structure and turbulence. In *Proceedings of the 2002 U.K. Workshop on Computational Intelligence*, pages 37–44, 2002. One citation in section 2.3.4.
- [39] Carlos M Fonseca and Peter J Fleming. Multiobjective genetic algorithms made easy: Selection, sharing, and mating restriction. *Proceedings of the 1st International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, pages 45–52, 1995. One citation in section 3.6.1.
- [40] Carlos M Fonseca and Peter J Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, 1995. 3 citations in sections 2.3.3, 2.3.4, and 2.4.
- [41] Carlos M Fonseca, Peter J Fleming, et al. Genetic algorithms for multiobjective optimization: Formulation discussion and generalization. In *ICGA*, volume 93, pages 416–423, 1993. One citation in section 2.3.4.
- [42] Guojun Gan, Chaoqun Ma, and Jianhong Wu. *Data clustering: theory, algorithms, and applications*, volume 20. Society for Industrial and Applied Mathematics, 2007. One citation in section 7.1.



- [43] S Gholizadeh and H Barati. A comparative study of three metaheuristics for optimum design of trusses. *International Journal of Optimization in Civil Engineering*, 3(3):423–441, 2012. One citation in section 2.2.
- [44] Ashish Ghosh and Satchidananda Dehuri. Evolutionary algorithms for multicriterion optimization: A survey. *International Journal of Computing & Information Sciences*, 2(1):38–57, 2004. One citation in section 8.2.
- [45] Crina Groşan and D Dumitrescu. A comparison of multiobjective evolutionary algorithms. *Acta Universitatis Apulensis*, 4, 2002. One citation in section 2.3.4.
- [46] Prabhat Hajela and C-Y Lin. Genetic search strategies in multicriterion optimal design. *Structural Optimization*, 4(2):99–107, 1992. One citation in section 2.3.1.
- [47] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3):107–145, 2001. One citation in section 7.6.
- [48] Kuk-Hyun Han and Jong-Hwan Kim. Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. *IEEE Transactions on Evolutionary Computation*, 6(6):580–593, 2002. One citation in section 1.2.
- [49] Julia Handl and Joshua Knowles. Improvements to the scalability of multiobjective clustering. In *In Proceedings of the 2005 IEEE Congress on Evolutionary Computation (IEEE CEC 2005)*, volume 3, pages 2372–2379, 2005. 2 citations in sections 7.6 and 7.6.
- [50] Julia Handl and Joshua Knowles. An evolutionary approach to multiobjective clustering. *IEEE Transactions on Evolutionary Computation*, 11(1):56–76, 2007. 3 citations in sections 7.3.1, 7.4.1, and 7.4.3.
- [51] SL Ho, Shiyong Yang, Guangzheng Ni, Edward WC Lo, and Ho-ching Chris Wong. A particle swarm optimization-based method for multiobjective design optimizations. *IEEE Transactions on Magnetics*, 41(5):1756–1759, 2005. One citation in section 2.3.4.

- [52] Jeffrey Horn, Nicholas Nafpliotis, and David E Goldberg. Multiobjective optimization using the niched pareto genetic algorithm. *IlliGAL Report*, (93005):61801–2296, 1993. One citation in section 2.3.4.
- [53] Jeffrey Horn, Nicholas Nafpliotis, and David E Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence*, pages 82–87. IEEE, 1994. One citation in section 2.3.4.
- [54] Eduardo R Hruschka, Ricardo José Gabrielli Barreto Campello, Alex Alves Freitas, and AC Ponce Leon F De Carvalho. A survey of evolutionary algorithms for clustering. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 39(2):133–155, 2009. 2 citations in sections 7.1 and 7.2.
- [55] Wei-Yen Hsu. Application of quantum-behaved particle swarm optimization to motor imagery EEG classification. *International Journal of Neural Systems*, 23(06), 2013. One citation in section 1.2.
- [56] Xiaohui Hu and Russell Eberhart. Multiobjective optimization using dynamic neighborhood particle swarm optimization. In *Proceedings of the World Congress on Computational Intelligence*, volume 2, pages 1677–1681. IEEE, 2002. One citation in section 2.3.2.
- [57] Xiaohua Huo, Lincheng Shen, and Huayong Zhu. A smart particle swarm optimization algorithm for multi-objective problems. In *Computational Intelligence and Bioinformatics*, pages 72–80. Springer, 2006. One citation in section 2.3.4.
- [58] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010. One citation in section 7.1.
- [59] Anil K Jain and Richard C Dubes. *Algorithms for clustering data*. Prentice-Hall, 1988. One citation in section 7.1.
- [60] Stefan Janson and Daniel Merkle. A new multi-objective particle swarm optimization algorithm using clustering applied to automated docking. In *Hybrid Metaheuristics*, pages 128–141. Springer, 2005. One citation in section 2.3.4.

- [61] Qing Jiang and Jian Li. A novel method for finding global best guide for multiobjective particle swarm optimization. In *Third International Symposium on Intelligent Information Technology Application, 2009. IITA 2009*, volume 3, pages 146–150. IEEE, 2009. 2 citations in sections 2.3.4 and 6.1.
- [62] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. Wiley, 2009. One citation in section 7.1.
- [63] J Kennedy and R C Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948, 1995. One citation in section 2.2.1.
- [64] James Kennedy, James F Kennedy, and Russell C Eberhart. *Swarm Intelligence*. Morgan Kaufmann, 2001. One citation in section 2.2.
- [65] Hajime Kita, Yasuyuki Yabumoto, Naoki Mori, and Yoshikazu Nishikawa. Multi-objective optimization by means of the thermodynamical genetic algorithm. In *Parallel Problem Solving From Nature-PPSN IV*, pages 504–512. Springer, 1996. One citation in section 4.4.
- [66] Joshua Knowles and David Corne. The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation. In *Proceedings of the 1999 Congress on Evolutionary Computation, 1999. (CEC 99)*, volume 1. IEEE, 1999. One citation in section 2.3.4.
- [67] Hsin-Chuan Kuo, Jiang-Ren Chang, Ching-Hsiang Liu, et al. Particle swarm optimization for global optimization problems. *Journal of Marine Science and Technology*, 14(3):170–181, 2006. One citation in section 1.2.
- [68] Xiaodong Li. A non-dominated sorting particle swarm optimizer for multiobjective optimization. In *Genetic and Evolutionary Computation-GECCO 2003*, pages 37–48. Springer, 2003. One citation in section 2.3.4.
- [69] Xiaodong Li. Better spread and convergence: Particle swarm multiobjective optimization using the maximin fitness function. In *Genetic and Evolutionary Computation Conference-GECCO 2004*, pages 117–128. Springer, 2004. 2 citations in sections 2.3.4 and 6.1.

- [70] HaiXia Long and XiuHong Zhang. Quantum-behaved particle swarm optimization based on comprehensive learning. In *Advances in Electronic Commerce, Web Application and Communication*, pages 15–20. Springer, 2012. One citation in section 1.2.
- [71] Ujjwal Maulik, Sanghamitra Bandyopadhyay, and Anirban Mukhopadhyay. *Multiobjective Genetic Algorithms for Clustering: Applications in Data Mining and Bioinformatics*. Springer, 2011. 6 citations in sections 7.1, 7.2, 7.3, 7.4.1, 7.4.3, and 7.6.
- [72] Giansalvatore Mecca, Salvatore Raunich, and Alessandro Pappalardo. A new algorithm for clustering search results. *Data & Knowledge Engineering*, 62(3):504–522, 2007. One citation in section 7.1.
- [73] Jerry M Mendel. Uncertain rule-based fuzzy logic system: introduction and new directions. 2001. One citation in section 2.3.2.
- [74] Sanaz Mostaghim and Jürgen Teich. The role of  $\varepsilon$ -dominance in multi objective particle swarm optimization methods. In *The 2003 Congress on Evolutionary Computation, 2003. CEC'03*, volume 3, pages 1764–1771. IEEE, 2003. One citation in section 2.3.4.
- [75] Sanaz Mostaghim and Jürgen Teich. Strategies for finding good local guides in multi-objective particle swarm optimization (mopso). In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium, 2003. SIS'03.*, pages 26–33. IEEE, 2003. 4 citations in sections 2.3.4, 3.2.1, 3.3, and 3.3.
- [76] Sanaz Mostaghim and Jürgen Teich. Covering pareto-optimal fronts by subswarms in multi-objective particle swarm optimization. In *Congress on Evolutionary Computation, 2004. CEC2004*, volume 2, pages 1404–1411. IEEE, 2004. One citation in section 2.3.4.
- [77] Sanaz Mostaghim and Jürgen Teich. Quad-trees: A data structure for storing pareto sets in multiobjective evolutionary algorithms with elitism. In *Evolutionary Multiobjective Optimization*, pages 81–104. Springer, 2005. One citation in section 6.1.
- [78] SN Omkar, Rahul Khandelwal, TVS Ananth, G Narayana Naik, and S Gopalakrishnan. Quantum behaved particle swarm optimization (qpso) for multi-objective

- design optimization of composite structures. *Expert Systems with Applications*, 36(8):11312–11322, 2009. 2 citations in sections 1.2 and 2.3.3.
- [79] Nikhil Padhye. Comparison of archiving methods in multi-objective particle swarm optimization (mopso): empirical study. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, pages 1755–1756. ACM, 2009. 5 citations in sections 2.3.4, 2.4, 6.1, 16, and 17.
- [80] Shan Pang, Hailin Zou, Wenchao Yang, and Zengfeng Wang. An adaptive mutated multi-objective particle swarm optimization with an entropy-based density assessment scheme. *Information & Computational Science*, 4:1065–1074, 2013. One citation in section 2.3.4.
- [81] Konstantinos E Parsopoulos and Michael N Vrahatis. Particle swarm optimization method in multiobjective problems. In *Proceedings of the 2002 ACM Symposium on Applied Computing*, pages 603–607. ACM, 2002. 5 citations in sections 1.2, 2.3.1, 2.3.2, 2.3.3, and 2.4.
- [82] Konstantinos E Parsopoulos and Michael N Vrahatis. Multi-objective particles swarm optimization approaches. *Multi-objective Optimization in Computational Intelligence: Theory and Practice*, pages 20–42, 2008. One citation in section 2.3.4.
- [83] Riccardo Poli, James Kennedy, and Tim Blackwell. Particle swarm optimization -an overview. *Swarm Intelligence*, pages 33–57, 2007. 2 citations in sections 2.2.1 and 2.2.1.
- [84] Kata Praditwong. *Evolutionary Multi-Objective Optimization Using Multiple Archives*. PhD thesis, University of Birmingham, June 2008. One citation in section 8.1.
- [85] Gregorio Toscano Pulido and Carlos A Coello Coello. A constraint-handling mechanism for particle swarm optimization. In *Proceedings of the Congress on Evolutionary Computation, 2004. CEC2004*, volume 2, pages 1396–1403. IEEE, 2004. One citation in section 2.3.4.
- [86] Gregorio Toscano Pulido and Carlos A Coello Coello. Using clustering techniques to improve the performance of a multi-objective particle swarm optimizer. In *Pro-*

- ceedings of the Genetic and Evolutionary Computation Conference–GECCO 2004*, pages 225–237. Springer, 2004. One citation in section 2.3.4.
- [87] Chafika Ramdane and Mohamed-Khireddine Kholadi. Improvements to the quantum evolutionary clustering. *International Journal of Data Analysis Techniques and Strategies*, 5(2):175–197, 2013. 3 citations in sections (document), 7.6.2, and 7.7.
  - [88] Chafika Ramdane, Souham Meshoul, Mohamed Batouche, and Mohamed-Khireddine Kholadi. A quantum evolutionary algorithm for data clustering. *International Journal of Data Mining, Modelling and Management*, 2(4):369–387, 2010. One citation in section 7.6.
  - [89] Carlo R Raquel and Prospero C Naval Jr. An effective use of crowding distance in multiobjective particle swarm optimization. In *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, pages 257–264. ACM, 2005. One citation in section 2.3.4.
  - [90] Tapabrata Ray and KM Liew. A swarm metaphor for multiobjective design optimization. *Engineering Optimization*, 34(2):141–153, 2002. One citation in section 2.3.4.
  - [91] Eréndira Rendón, Itzel Abundez, Alejandra Arizmendi, and Elvia M Quiroz. Internal versus external cluster validation indexes. *International Journal of Computers and Communications*, 5(1):27–34, 2011. One citation in section 7.6.
  - [92] Margarita Reyes-Sierra and CA Coello Coello. Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *International journal of Computational Intelligence Research*, 2(3):287–308, 2006. 5 citations in sections 1.2, 2.2.1, 2.2.2, 2.3.4, and 2.4.
  - [93] Dian Palupi Rini, Siti Mariyam Shamsuddin, and Siti Sophiyati Yuhaniz. Particle swarm optimization: technique, system and challenges. *International Journal of Computer Applications*, 14(1):19–26, 2011. 3 citations in sections 1.2, 2.2.1, and 2.2.1.
  - [94] Dhish Kumar Saxena, Joao A Duro, Ashutosh Tiwari, Kalyanmoy Deb, and Qingfu Zhang. Objective reduction in many-objective optimization: Linear and nonlinear

- algorithms. *Evolutionary Computation, IEEE Transactions on*, 17(1):77–99, 2013. 2 citations in sections 2.4 and 8.2.
- [95] James David Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 93–100. L. Erlbaum Associates Inc., 1985. 2 citations in sections 2.3.3 and 3.6.1.
- [96] Matthew Settles. An introduction to particle swarm optimization. *Department of Computer Science, University of Idaho*, 2005. 2 citations in sections 1.2 and 2.2.1.
- [97] Yuhui Shi and Russell Eberhart. A modified particle swarm optimizer. In *In Proceedings of Evolutionary Computation*, pages 69–73. IEEE, 1998. 2 citations in sections 2.2.1 and 2.2.1.
- [98] Yuhui Shi and Russell C Eberhart. Parameter selection in particle swarm optimization. In *Evolutionary Programming VII*, pages 591–600. Springer, 1998. 2 citations in sections 2.2.1 and 2.2.1.
- [99] Margarita Reyes Sierra and Carlos A Coello Coello. Improving PSO-based multi-objective optimization using crowding, mutation and  $\varepsilon$  dominance. In *Evolutionary Multi-criterion Optimization*, pages 505–519. Springer, 2005. 2 citations in sections 2.3.4 and 8.2.
- [100] Prisadarng Skolpadungket, Keshav Dahal, and Napat Harnpornchai. Portfolio optimization using multi-objective genetic algorithms. In *IEEE Congress on Evolutionary Computation, 2007. (CEC 2007)*, pages 516–523. IEEE, 2007. One citation in section 2.3.4.
- [101] Nidamarthi Srinivas and Kalyanmoy Deb. Multiobjective optimization using non-dominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994. 2 citations in sections 2.3.4 and 4.4.
- [102] Benno Stein, Sven Meyer zu Eissen, and Wißbrock Frank. On cluster validity and the information need of users. *Third IASTED International Conference on Artificial Intelligence and Applications*, pages 216–221, 2003. One citation in section 7.6.

- [103] Jun Sun, Wei Fang, Vasile Palade, Xiaojun Wu, and Wenbo Xu. Quantum-behaved particle swarm optimization with gaussian distributed local attractor point. *Applied Mathematics and Computation*, 218(7):3763–3775, 2011. One citation in section 3.2.1.
  
- [104] Jun Sun, Wei Fang, Xiaojun Wu, Vasile Palade, and Wenbo Xu. Quantum-behaved particle swarm optimization: Analysis of individual particle behavior and parameter selection. *Evolutionary Computation*, 20(3):349–393, 2012. 6 citations in sections 1.2, 2.2.2, 2.2.2, 3.2.3, 3.6.3, and 4.4.1.
  
- [105] Jun Sun, Bin Feng, and Wenbo Xu. Particle swarm optimization with particles having quantum behavior. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 325–331, 2004. 6 citations in sections 1.2, 2.2.2, 2.2.2, 3.1, 3.6.3, and 4.4.1.
  
- [106] Jun Sun, Wenbo Xu, and Bin Feng. A global search strategy of quantum-behaved particle swarm optimization. In *IEEE Conference on Cybernetics and Intelligent Systems*, pages 111–116, 2004. One citation in section 1.2.
  
- [107] Masahiro Tanaka, Hikaru Watanabe, Yasuyuki Furukawa, and Tetsuzo Tanino. GA-based decision support system for multicriteria optimization. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics-2*, volume 2, pages 1556–1561. IEEE, 1995. One citation in section 4.4.
  
- [108] Mehdi Tavakolan and Babak Ashuri. Comparison of evolutionary algorithms in non-dominated solutions of time-cost-resource optimization problem. In *Proceedings of the 48th ASC Annual International Conference*, 2012. One citation in section 2.2.2.
  
- [109] Gregorio Toscano-Pulido, Carlos A Coello Coello, and Luis Vicente Santana-Quintero. EMOPSO: A multi-objective particle swarm optimizer with emphasis on efficiency. In *Evolutionary Multi-Criterion Optimization*, pages 272–285. Springer, 2007. One citation in section 2.3.4.
  
- [110] Luca Trevisan. Lecture notes on computational complexity. *Notes written in Fall*, 2002. One citation in section 2.1.4.



- [111] R Umarani and V Selvi. Particle swarm optimization-evolution, overview and applications. *International Journal of Engineering Science and Technology*, pages 2802–2806, 2010. 5 citations in sections 1.2, 2.2, 2.2.1, 2.2.1, and 2.2.1.
- [112] David A Van Veldhuizen. Multiobjective evolutionary algorithms: classifications, analyses, and new innovations. Technical report, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1999. 2 citations in sections 2.1.2 and 2.3.4.
- [113] Mario Alberto Villalobos-Arias, Gregorio Toscano Pulido, and Carlos A Coello Coello. A proposal to use stripes to maintain diversity in a multi-objective particle swarm optimizer. In *Proceedings of the IEEE Swarm Intelligence Symposium, 2005. SIS 2005*, pages 22–29. IEEE, 2005. One citation in section 2.3.4.
- [114] Upali Wickramasinghe and Xiaodong Li. Choosing leaders for multi-objective pso algorithms using differential evolution. In *Simulated Evolution and Learning*, pages 249–258. Springer, 2008. One citation in section 2.3.4.
- [115] Xuanli Lisa Xie and Gerardo Beni. A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8):841–847, 1991. One citation in section 7.3.1.
- [116] Rui Xu, Donald Wunsch, et al. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005. One citation in section 7.1.
- [117] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000. 2 citations in sections 2.3.4 and 3.6.1.
- [118] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. In *Proceedings of Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*. Athens, Greece, 2002. 2 citations in sections 2.3.4 and 6.1.
- [119] Eckart Zitzler and Lothar Thiele. *An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach*. Citeseer, 1998. One citation in section 3.6.1.

- [120] Eckart Zitzler and Lothar Thiele. Multiobjective optimization using evolutionary algorithms- a comparative case study. In *Parallel Problem Solving from Nature-PPSN V*, pages 292–301. Springer, 1998. 2 citations in sections 2.3.1 and 2.3.4.
- [121] Albert Y Zomaya. *Handbook of nature-inspired and innovative computing: integrating classical models with emerging technologies*. Springer, 2006. One citation in section 1.1.